EXHIBIT B

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
60/760,362	01/06/2006	David Brown	25000-11134

CONFIRMATION NO. 9466

758
FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

MISCELLANEOUS NOTICE

OC000000040702600

Date Mailed: 03/19/2010

A communication which cannot be delivered in electronic form has been mailed to the applicant.

Doc Code: N572

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
60/760,362	01/06/2006	David Brown	25000-11134

CONFIRMATION NO. 9466

OC000000040702600

758
FENWICK & WEST LLP
SILICON VALLEY CENTER
801 CALIFORNIA STREET
MOUNTAIN VIEW, CA 94041

Cc: PATENT LAW WORKS/PROXENSE
165 SOUTH MAIN ST
SALT LAKE CITY, UT 84111

Date Mailed: 03/19/2010

DENIAL OF REQUEST FOR POWER OF ATTORNEY

The request for Power of Attorney filed 03/12/2010 is acknowledged. However, the request cannot be granted at this time for the reason stated below.

□ The Power of Attorney you provided did not comply with the new Power of Attorney rules that became effective on June 25, 2004. See 37 CFR 1.32.

□ The revocation is not signed by the applicant, the assignee of the entire interest, or one particular principal attorney having the authority to revoke.

☑ The Power of Attorney is from an assignee and the Certificate required by 37 CFR 3.73(b) has not been received.

□ The person signing for the assignee has omitted their empowerment to sign on behalf of the assignee.

□ The inventor(s) is without authority to appoint attorneys since the assignee has intervened as provided by 37 CFR 3.71.

□ The signature(s) of ______________________, a co-inventor in this application, has been omitted. The Power of Attorney will be entered upon receipt of confirmation signed by said co-inventor(s).

□ The person(s) appointed in the Power of Attorney is not registered to practice before the U.S. Patent and Trademark Office.

Questions relating to this Notice should be directed to the Application Assistance Unit.

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

PTO/SB/80 (11-08)
Approved for use through 11/30/2011. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY TO PROSECUTE APPLICATIONS BEFORE THE USPTO

I hereby revoke all previous powers of attorney given in the application identified in the attached statement under 37 CFR 3.73(b).

I hereby appoint:

☑ Practitioners associated with the Customer Number: 89194

OR

□ Practitioner(s) named below (if more than ten patent practitioners are to be named, then a customer number must be used):

Name	Registration Number	Name	Registration Number

as attorney(s) or agent(s) to represent the undersigned before the United States Patent and Trademark Office (USPTO) in connection with any and all patent applications assigned only to the undersigned according to the USPTO assignment records or assignment documents attached to this form in accordance with 37 CFR 3.73(b).

Please change the correspondence address for the application identified in the attached statement under 37 CFR 3.73(b) to:

☑ The address associated with Customer Number: 89194

OR

□ Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

Assignee Name and Address:

Proxense, LLP
689 NW Stonepine Drive
Bend, OR 97710-6818

A copy of this form, together with a statement under 37 CFR 3.73(b) (Form PTO/SB/96 or equivalent) is required to be filed in each application in which this form is used. The statement under 37 CFR 3.73(b) may be completed by one of the practitioners appointed in this form if the appointed practitioner is authorized to act on behalf of the assignee, and must identify the application in which this Power of Attorney is to be filed.

SIGNATURE of Assignee of Record

The individual whose signature and title is supplied below is authorized to act on behalf of the assignee

Signature		Date	2/22/10
Name	John Giobbi	Telephone	541-382-5745
Title	Chief Executive Officer		

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Acknowledgement Receipt

EFS ID:	7197576
Application Number:	60760362
International Application Number:	
Confirmation Number:	9466
Title of Invention:	Securing transactions between an electric key and lock within proximity of each other
First Named Inventor/Applicant Name:	David Brown
Customer Number:	00758
Filer:	Elizabeth D. Ruzich/Esther Kim
Filer Authorized By:	Elizabeth D. Ruzich
Attorney Docket Number:	25000-11134
Receipt Date:	12-MAR-2010
Filing Date:	06-JAN-2006
Time Stamp:	18:40:55
Application Type:	Provisional

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Power of Attorney	ProxensePowerofAttorney.pdf	1014805 d19ae545af83ee177fab166e3f05eaf890f8db2a	no	1

Warnings:

Information:

Total Files Size (in bytes):	1014805

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Please type a plus sign (+) inside this box→ +

+

modified
PTO/SB/16(8-00)

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION under 37 CFR 1.53(c).

Docket Number:	25000-11134

INVENTOR(s)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City And Either State Or Foreign Country)
David	Brown	Jupiter, FL
Fred	Hirt	Brookfield, IL

□ Additional inventors are being named on ___ separately numbered sheets attached hereto.

TITLE OF THE INVENTION (500 characters max.)

SECURING TRANSACTIONS BETWEEN AN ELECTRIC KEY AND LOCK WITHIN PROXIMITY OF EACH OTHER

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☒ Customer Number **00758**

ENCLOSED APPLICATION PARTS (check all that apply)

☒ Specification	*No. of Pages:*	176	☒ Return Postcard	
□ Drawing(s)	*No. of Sheets:*		□ CD(s), Number	
□ Application Data Sheet See 37 CFR 1.76			□ Other *(specify)*	

METHOD OF PAYMENT (check all that apply)

☒ Applicant claims small entity status. See 37 CFR 1.27

☒ Fee Transmittal Form Enclosed (in duplicate) ☒ Check Enclosed

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

□ Yes, the name of the U.S. Government Agency and the Government contract number are: ______________________

SIGNATURE OF ATTORNEY OR AGENT

Signature:			
Attorney/Reg. No.:	Brian M. Hoffman, Reg. No. 39,713	Dated:	January 6, 2006

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10 in an envelope addressed to: Mail Stop Provisional Patent Application, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Signature:			
Typed or Printed Name:	**Brian M. Hoffman**	Dated:	January 6, 2006
Express Mail Mailing Number:	**EV301187035US**		

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act. 2005 (H.R. 4818).

FEE TRANSMITTAL
For FY 2006

☒ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT ($) 350.00

Complete if Known	
Application Number	Not yet known
Filing Date	January 6, 2006
First Named Inventor	David Brown
Examiner Name	Not applicable
Art Unit	Not applicable
Attorney Docket No.	25000-11134

METHOD OF PAYMENT (check all that apply)

☒ Check ☐ Credit Card ☐ Money Order ☐ Other (please identify):

☐ Deposit Account Deposit Account Number: 19-2555 Deposit Account Name: Fenwick & West LLP

☒ Charge all required fee(s) or any underpayment of fee(s) ☒ Credit any overpayments

due under 37 CFR §1.16 or §1.17 during the pendency of this application

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
		Small Entity		Small Entity		Small Entity	
Application Type	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fee ($)	Fees Paid ($)
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	100

2. EXCESS CLAIM FEES

		Small Entity
Fee Description	Fee	Fee
Each claim over 20 or, for Reissues, each claim over 20 and more than in the original patent	50	25
Each independent claim over 3 or, for Reissues, each independent claim more than in the original patent	200	100
Multiple dependent claims	360	180

Total Claims **Extra Claims** **Fee ($)** **Fee Paid ($)** **Multiple Dependent Claims**
Fee ($) **Fee Paid ($)**

- 20 or HP = x =

HP = highest number of total claims paid for, if greater than 20

Total Independent Claims **Extra Claims** **Fee ($)** **Fee Paid ($)**

- 3 or HP = x =

HP = highest number of independent claims paid for, if greater than 3

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		Extra Sheets		Number of each additional 50 or fraction thereof	Fee ($)		Fee Paid ($)
176	- 100 =	76	/ 50 =	(round up to a whole number) 2x	125	=	250

4. OTHER FEE(S)

Fees Paid ($)

Non-English Specification, $130 fee (no small entity discount)
Assignment:
Other:

SUBMITTED BY

Signature		Registration No.: 39,713 (Attorney/Agent)	Telephone: (415) 875-2484
Name (Print/Type)	Brian M. Hoffman		Date: January 6, 2006

PROXIMITY

PROXENSE

TECHNOLOGIES

TruProx® Specification

Table of Contents

List of Figures

List of Tables

1 Acronyms

802.15.4	An IEEE-SA communications specification for wireless personal area networks
CRDC	Coordinator RDC, an RDC placed in "coordination" mode which acts to prevent wireless collisions and to greatly extend the battery life of individual PDK® units
FOB	Generic name for the Proxense PDK®
IEEE	The Institute of Electrical and Electronic Engineers
IEEE-SA	The Institute of Electrical and Electronics Engineers Standards Association
ISO	International Standards Organization
ITU	International Telecommunications Union
ITU-T	The Telecommunications division of the ITU which produces standards
lsb	Least significant bit
LSB	Least significant byte
MAC	Message Authentication Code, a checksum-like code used to verify that an encrypted message has not been altered during transmission
MAC layer	Media Access Control layer, the protocol that controls access to the physical transmission medium in a network, synonymous with the data link layer in the OSI model.
msb	Most significant bit
MSB	Most significant by
OSI	Open Systems Interconnect, a joint ISO and ITU-T standard for computer networks and protocols
PDK®	Personal Digital Key, the Proxense device carried by the user allowing both tracking and digital remote access of various applications, games, and services
POD	Generic name for the Proxense PDK®
RDC	Reader Decoder Circuit, the Proxense device which senses the distance of and interacts with the Proxense PDK® to link the user to various applications, games, and services
RF	Radio Frequency, a transmission medium based on electromagnetic propagation
TruProx®	True Proximity, a proximity-based personal digital key system developed by Proxense
Zigbee	A low power, low data rate wireless network protocol based on the IEEE 802.15.4 specification, named after the zigzag flight characteristic of a bee

2 References

Ref 1: IEEE STD 802.15.4-2003,

3 General Overview

This document will provide a functional description of the Proxense TruProx® specifications. It details the TruProx architecture and its components: the Personal Digital Key (PDK®) architecture, Reader Decoder Circuit (RDC) architecture, the communications protocol, and various system configurations.
Throughout this document, the Proxense PDK® is also referred to informally as the "POD" or "FOB".

The Proxense TruProx system and intellectual property together provide a method of securing transactions between a user carrying a Proxense PDK®, or "electronic key" and a Proxense fixed part, the RDC, which are within a reasonable proximity of each other.

Since the Proxense TruProx system is also based on proximity detection, it inherently provides location and tracking information on users within the system.

3.1 TruProx® Basic System Architecture

This section will give a brief description of the various architectures the Proxense TruProx system will be used in. The descriptions are basic concepts of how an RDC and PDK®, a.k.a. POD or FOB, communicate and how multiple RDCs and PDK®s coexist within a given geographical area.

3.1.1 Single Cell Operation of the RDC and PDK®

A Proxense system in the most basic form is comprised of a PDK® and an RDC unit. The PDK® or FOB is a device that can be worn, placed in the pocket of a user, or attached to equipment, has a bidirectional wireless communications transceiver, and contains both public and secret electronic ID numbers as well as cryptographic keys.

The RDC is a device providing a fixed access point for a mobile PDK or POD to communicate with. The RDC is affectively a gatekeeper for a POD that wants to access the system. The RDC may be incorporated into a casino floor, electronic game, doorway, pedestrian traffic monitoring point, or into a personal computer application, an e-commerce device such as ATM machines, or any other application where secure transactions must take place.

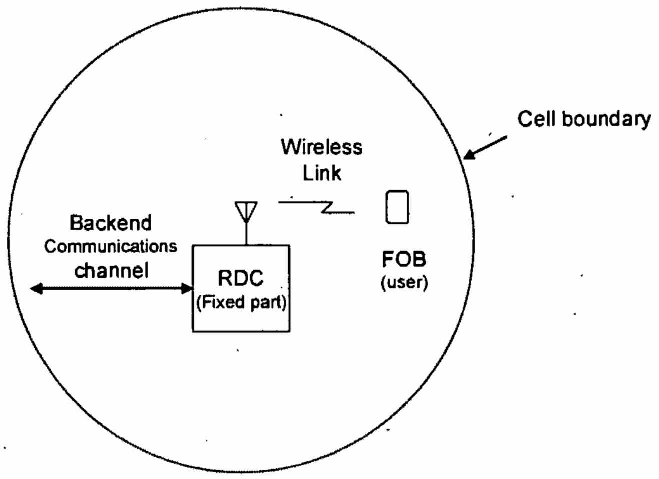


Figure 1: Basic Proxense System (single cell)

As shown in Figure 1, The RDC is a fixed device that has a cell radius defined by its RF coverage boundary. When a user carrying a POD comes into proximity of the RDC by entering the cell's RF coverage area, or cell boundary, a communications session begins between the POD and the RDC via the wireless link. If the RDC finds that the POD is authorized to communicate, information between the POD and the RDC are securely exchanged. Information securely extracted from the user's pod can then be used locally or can be sent through a backend communications channel to a central server.

When the transaction completes or when the POD leaves the RDC's cell boundary communications ceases. The RDC remains in idle, i.e. tracking, mode until another POD enters its cell boundary.

3.1.2 Unsynchronized Multi-Cell Operation of Multiple RDC and PDK®

In some system deployment configurations multiple RDC cells can exist in a confined area. The RDCs may or may not be aware of each other, but all still must be able to interact with the POD devices. The POD devices in turn must also interact with the RDCs.

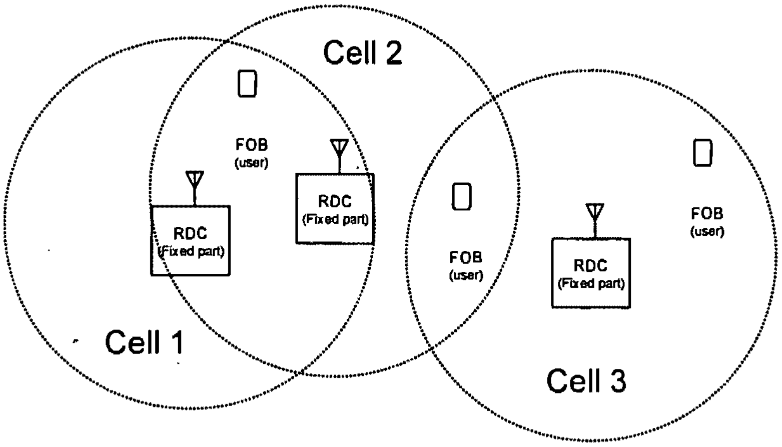


Figure 2: Unsynchronized Multi-cell configuration

Consider the case shown in Figure 2 above where three partially overlapping RDC cells exist. Each RDC is independent and has no association with the other RDCs. Although some cell overlap exists, each RDC must be capable of communicating to any PODs that are within its cell boundary.

Based on the TruProx® wireless protocol, each RDC is capable of determining if energy is present on any given channel. The RDC can then determine the best channel to operate on and continue to place an identification marker out for any POD that enters its cell boundary.

The POD itself is responsible for locating an RDC by looking through the available channels, communicating to an RDC, and notifying it of its presence. In the case where two RDCs can receive communications from (herein after referred to as "see") each other (Cells 1 & 2 above) each RDC is capable of frequency planning.

In the case where the cells overlap, but each RDC cannot directly communicate with the other – as in the case of Cell 2 and Cell 3 above – any POD intending to access an RDC, must alert the RDC of possible collisions on the Channel which the RDC is operating.

3.1.3 Synchronized Multi-Cell Operation of Multiple RDC and PDK®

With a minimum of organization in a multi-cell system, RDCs can be placed to allow an overlap of cells between each adjacent RDC within a confined area. This permits each RDC to be aware of its surrounding RDCs allowing synchronization of each RDC to the other.

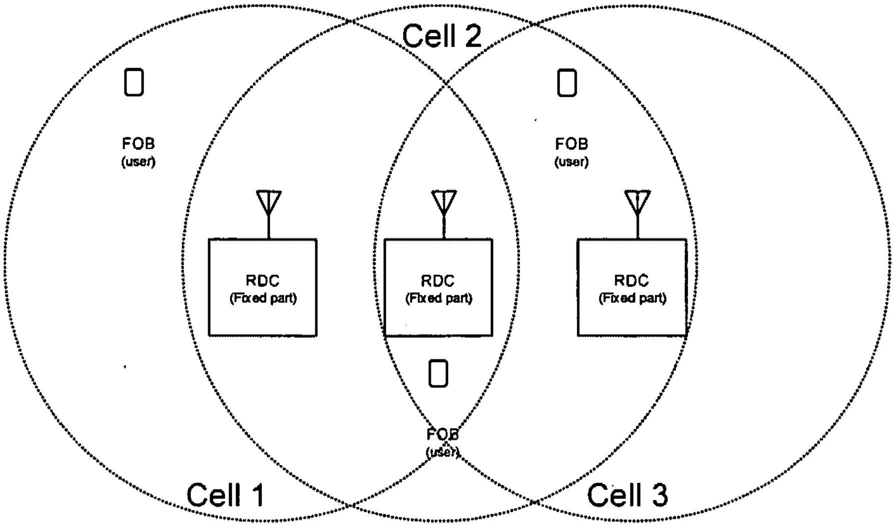


Figure 3: Synchronized Multi-Cell

One example of synchronized multi-call operation is depicted in Figure 3 where three RDCs are shown with each RDC's cell boundary overlapping the adjacent RDC's. In this configuration, each RDC can initiate communication to an adjacent RDC. This begins the negotiation process between RDCs to determine which RDC is to be the coordinator.

Although any RDC can be a coordinator, in this example the RDC located in cell 2 would be a prime candidate. Its selection permits ubiquitous coverage of the RDCs in the geographic area shown, additionally providing multiple transactions and timing alignment through the daisy chain (i.e. RDC 1 synchronizes RDC 2, and RDC 2 synchronizes RDC 3).

Each RDC may also share frequency and timeslot information between each other and with the PODs. The only remaining issue concerns PODs at the outer boundaries. If a POD is located on the outer skirts of a non-overlapping RDC cell, the POD may still monitor the other channels that the adjacent RDCs are on, but may not have access to these RDCs. Note that this would cause additional battery life to be consumed when it is not necessary.

Although in an unsynchronized system, as described in section 3.1.2, a POD can communicate just as effectively to an RDC as described in this section, the POD cannot conserve energy as efficiently as in this configuration. Additionally, as cell density increases, more collisions begin occurring and active communications times increase.

3.1.4 TruProx® Coordinated Multi-Cell Operation

The coordinated multi-cell system provides ubiquitous POD and RDC synchronization as well as battery conservation within the system. This is the preferred mode of operation for devices in the system deployment. In addition to ubiquitous synchronization, channel and frequency capacity can both be coordinated thus reducing collisions while increasing system throughput.

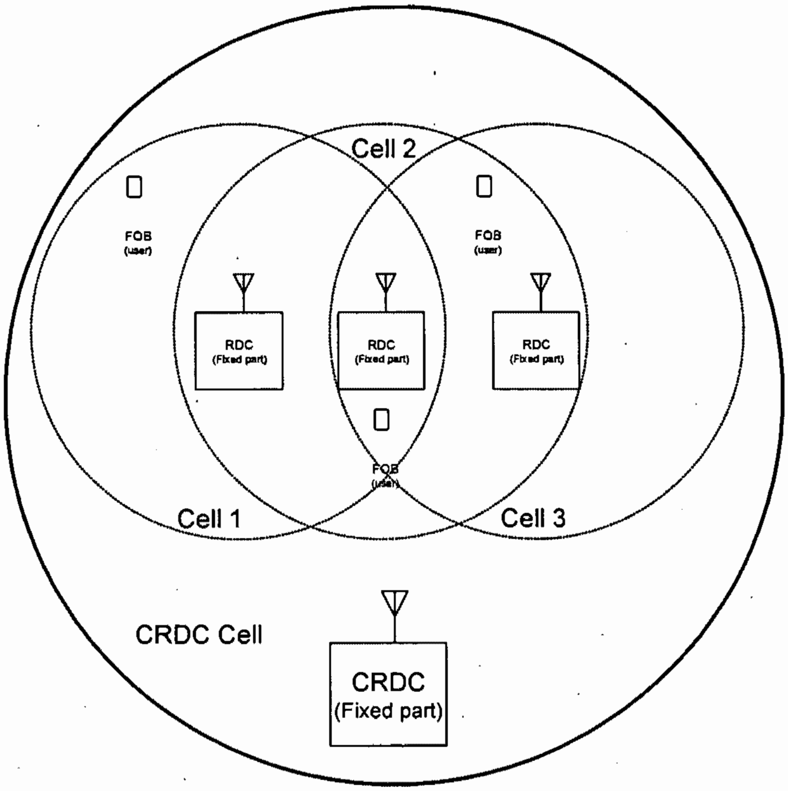


Figure 4: Coordinated Multi-Cell

As shown in Figure 4, a **TruProx® Coordinator RDC** (CRDC) has ubiquitous coverage to all cells in the system. By providing wide-area coverage *all devices*, both PODs and RDCs, in the coverage area are able to monitor a CRDC wireless transmission beacon and determine how and when to communicate to PODs located within their geographic area.

The CRDC provides periodic beacon transmissions which provide synchronization to all devices within the coverage area. In addition, the CRDC provides additional information such as:

- System timing
  - Beacon transmit rate
  - Framing information
- Channel information
- System identification
- Cluster identification

Note that although the CRDC provides the timing and certain system related information, the RDC and POD are still capable of communicating between themselves.

4 Device Overview

As previously mentioned, there are 3 types of devices that operate within the Proxense system. They are:

- The PDK® or POD – a device carried by the user and contains secret electronic keys.
- The RDC – a device that acts as the gatekeeper to the system and communicates with the POD.
- The CRDC – a device that coordinates the system and provides synchronization for all RDCs and PODs in a geographic area.

A basic description and architecture of each of these devices will follow.

4.1 POD

The POD is a device which is physically the size of an automotive remote entry device attached to a keychain. Based on the specific application the POD resides in, the POD may have different configurations to interact with the user. Some of the configurations are:

- No user input or display
- Single button press user input
- Multi-button user input
- Biometric input

- Interactive user input and display.

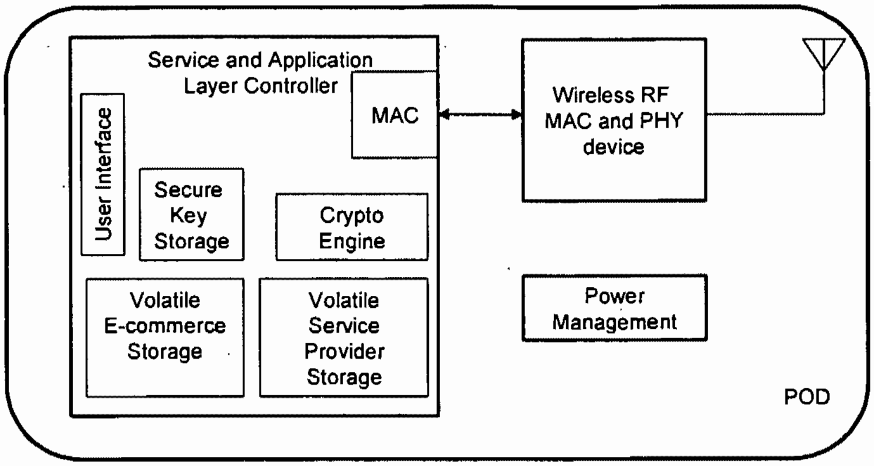


Figure 5: Basic POD Architecture

As shown in Figure 5, the POD consists of a wireless Radio Frequency (RF) Medium Access Control (MAC) and Physical (PHY) layer device which provides bidirectional communications to outside RF wireless devices such as the RDC.

The Wireless RF MAC and PHY device for the first generation product will be based on the IEEE 802.15.4 specification.

The Service and Application Layer controller includes portions of the MAC that are specific to the Proxense system wireless protocol. The Service and Application layers are also specific to the Proxense system and provide the specific functions used to protect the electronic keys and services.

The Service and Applications controller contains a user interface, which if used, will allow user interaction with the device. Such interaction could be a single button, where a button press can awake the unit or cause a transaction to take place.

A crypto engine is also present which implements portions of Proxense intellectual property.

Three different storage elements also exist within the controller and are shown below in Figure 6.

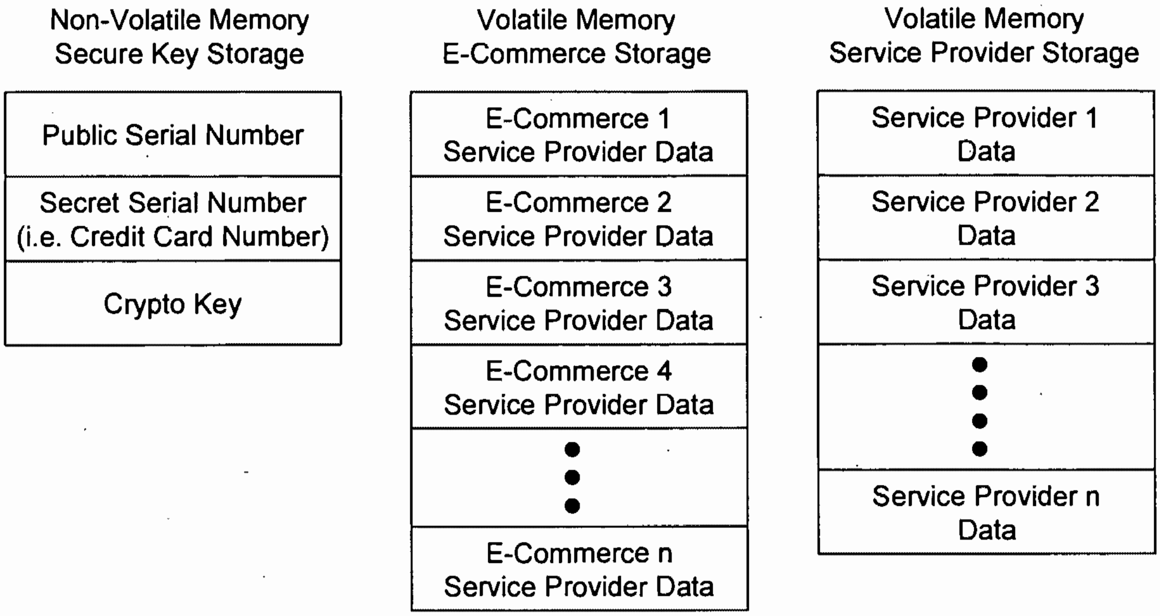


Figure 6: POD Secure Memory Map example

Referring to Figure 6, the three primary memory areas related to security which is only accessible with an RDC containing the appropriate security information, E-commerce financial institutions, and private service providers.

The 3 primary areas are designated as follows:

- Non-volatile memory Secure Key Storage – The information stored in this area is programmed at manufacturing time of the controller and except for the Public Serial Number, the information remains secure and is not readable by any outside RDCs.

  The Public Serial Number is used to identify the POD and to allow secure lookup of the Secure Serial Number and Crypto Key via a remote server.

  The Secure Serial Number is equivalent to a users' credit card number and is not visible to the outside world.

  The Crypto Key is used to allow decode of the secure serial number from the RDC.

- Volatile Memory E-commerce Storage – The information stored in this area will be specific for each financial institution wanting to use the POD for financial transactions.

  Once the E-Commerce financial institution is able to access the key by knowing the Secret Key in non-volatile memory, the financial institution via the POD must

send the E-Commerce Service identification to the POD, and if correct, the POD will share that E-Commerce secret key with the provider.
Essentially the E-commerce Service Identification is the enabler for access to the E-commerce secret key. The E-Commerce Secret Key is another tier of protection related only to one E-Commerce service provider.

Multiple E-Commerce Service ID's and Keys can be stored in the POD.

- Volatile Memory Service Provider Storage – This area is intended for private sector security and allows a service provider in the private sector to store secret keys and other secure information such as privileges in the POD. A Service Provider POD identification is stored to allow the service provider to easily identify the user. In addition, a Service Provider Service Identification value is stored and is used to only allow that service provider to access that information. The POD validates the Service Provider Identification via the Secret key before allowing access that Service Provider portion of the POD.

  The service provider information area is of variable length and allows a service provider the flexibility to store various parameters. The length is determined by a byte count following the Service Providers Secret Key in the memory area as shown in Figure 7.

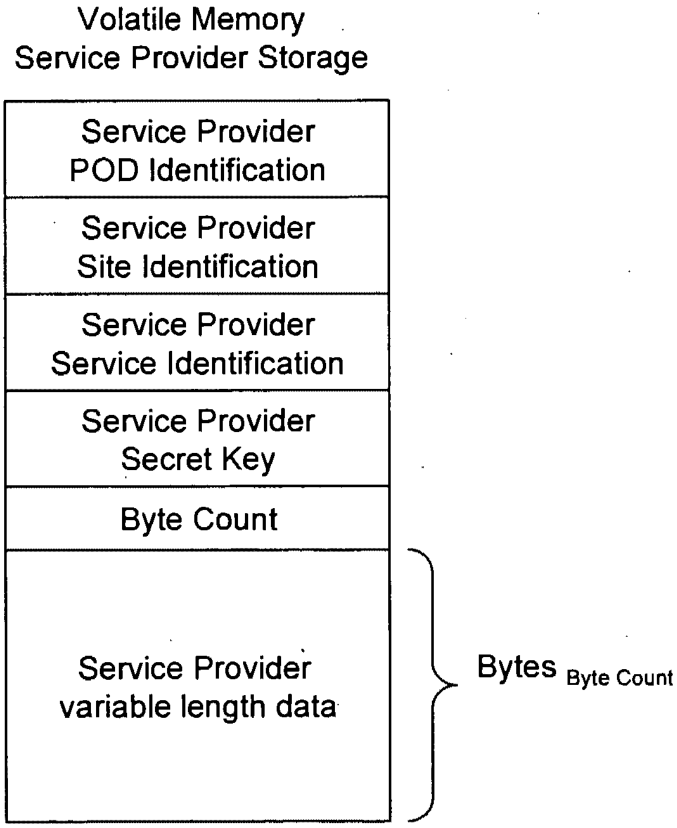

Volatile Memory
Service Provider Storage
Service Provider POD Identification
Service Provider Site Identification
Service Provider Service Identification
Service Provider Secret Key
Byte Count
Service Provider variable length data
Bytes Byte Count

Figure 7: Service Provider Memory area instance

The information stored within the Service Provider variable length data will be defined by the service provider.

4.2 RDC

The RDC is the fixed portion of the system and is used to allow POD access into a private system or financial institution. The RDC may have different configurations to support different secure transactions for a service provider. Some examples of where the RDC is used are:

- Personal Computers and disk drives
- Automatic Teller Machines
- Casino Slot Machines
- User / equipment location tracking

In many cases the RDC acts as a gatekeeper only allowing authorized individuals access to specific information or transactions. In other cases, since the RDC uses proximity for determining if a POD is in a geographical area, the RDC can also be used for tracking of PODs within a given area or network.

The primary RDC device may be developed in to different communications configurations. The First configuration will only use a single Wireless RF MAC and PHY device, where the second configuration will incorporate a dual Wireless RF MAC and PHY device.

4.2.1 RDC with Single PHY

The first configuration will be the primary platform for almost all Proxense security purposes, but the second configuration will allow easy synchronization with some additional networking functionality for advanced systems.

The system shown in Figure 8 is of the first configuration and requires all communications to be passed through a single Wireless RF MAC and PHY device.

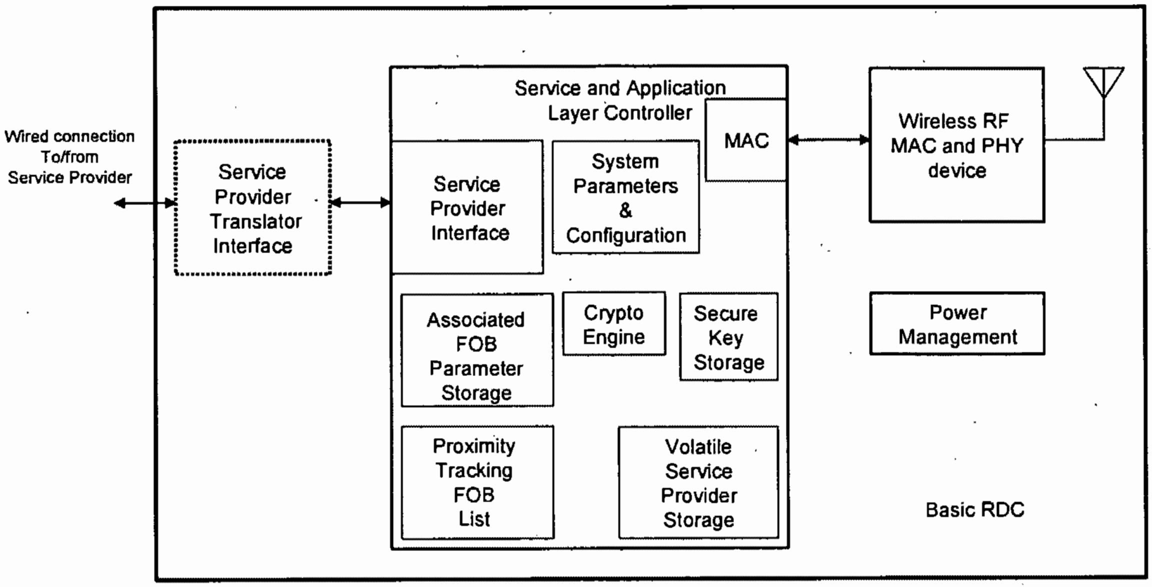

Wired connection
To/from
Service Provider
Service Provider Translator Interface
Service and Application Layer Controller
MAC
Service Provider Interface
System Parameters & Configuration
Associated FOB Parameter Storage
Crypto Engine
Secure Key Storage
Proximity Tracking FOB List
Volatile Service Provider Storage
Wireless RF MAC and PHY device
Power Management
Basic RDC

Figure 8: Basic RDC Configuration

As shown in Figure 5, the RDC consists of a wireless Radio Frequency (RF) Medium Access Control (MAC) and Physical (PHY) layer device which provides bidirectional communications to outside RF wireless devices such as the POD.

The Wireless RF MAC and PHY device for the first generation product will be based on the IEEE 802.15.4 specification.

Although not apparent by the diagram, the antenna structure for the RDC will vary based on the application it is used in.

The Service and Application Layer controller includes portions of the MAC that are specific to the Proxense system wireless protocol. The Service and Application layers are also specific to the Proxense system and provide several functions related to associating and tracking PODs as well as providing information back to the service provider.

The Service and Applications controller contains System Parameters and Configuration information used to identify the RDC and define how the RDC operates within a given environment. System Parameters and Configuration information define how the RF link is time slotted and how RF frequencies are used in the system. These values are optimized to reduce power consumption within the POD device.

A crypto engine is also present which implements portions of Proxense intellectual property.

Several storage elements also exist within the controller. The Secure Key Storage area is programmed at the factory and defines public, secret, and crypto keys for the RDC.

There are three other memory areas in the controller which dynamically change due to system changes and wireless POD connections.

The Volatile Service Provider Storage allows the service provider to store semi-static information related to a specific POD or group of PODs for real time access for those devices. An example would be a hotel room door. With a hotel room door, the service provider information related to the POD can be stored in the RDC. When a user becomes within proximity of the door, the door could unlock. In this scenario, the RDC has not required to interface with a backend server in real time reducing the bandwidth to the backend server, while allowing the user immediate access.

The Proximity Tracking POD List contains POD identity, signal quality metrics, and timestamps for each POD that is in proximity of the RDC. This information is not specifically used in the RDC to perform an operation, but is relayed to the backend server where location tracking is employed.

The Associated POD Parameter Storage contains a list of one or more PODs actively performing transactions with the RDC. Although the transactions are performed with the RDC, the actual processing result from the RDC to/from POD transaction is passed to the backend server for further processing.

The Service Provider Interface allows both control and query of the RDC device as well as provides the transport for the keys from the POD. The Service Provider Interface will use a standard interface, such as SPI, I2C, UART, etc. and will allow control and status of the RDC in its most primitive form.

The Service Provider Translator Interface may physically reside on board with these circuits. The Service Provider Translator Interface allows for easy adoption to various physical interfaces and protocols without having to alter the Proxense System and Application Controller.

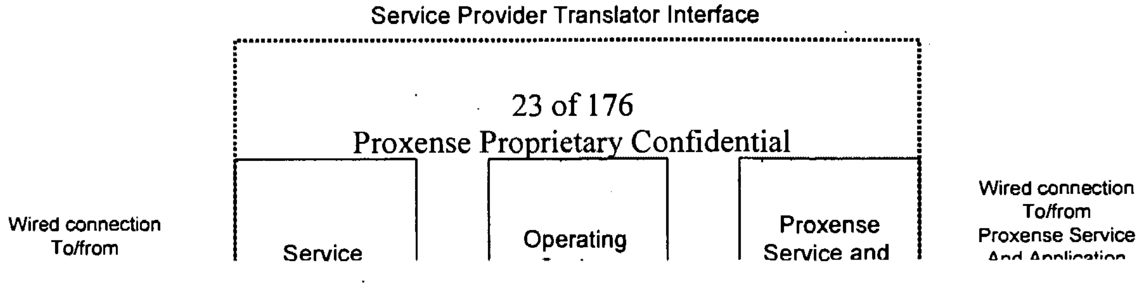

Service Provider Translator Interface
Wired connection To/from
Service
Operating
Proxense Service and
Wired connection To/from Proxense Service And Application

Figure 9: Service Provider Translator Interface

As shown in Figure 9, there are 3 primary blocks associated with the Service Provider Translator Interface. There are 2 Application Program Interfaces (APIs) with one for the Proxense Service and Application Controller Interface, and one for the Service Provider Interface. An operating system and protocol translator sits between the 2 APIs and provides the necessary functions to pass data and control to each side in an understandable fashion.

The 2 APIs are effectively function calls to the Operating system. The Protocol translator assists the API in properly passing the data and performing the necessary framing for the specific interface used at each end.

By providing the Service Provider Translator Interface, the Service Provider is not required to know the details of Proxense technology and timing and therefore reduces development time.

4.2.2 RDC with Dual PHY

The second configuration will be the alternate platform used for high density location tracking and secure communications. The intention of the second configuration is to allow one Wireless RF MAC and PHY device to maintain synchronization with a CRDC, and have the ability to pass networking related information, while the second Wireless RF MAC and PHY device communicates to the PODs within the RDCs cell.

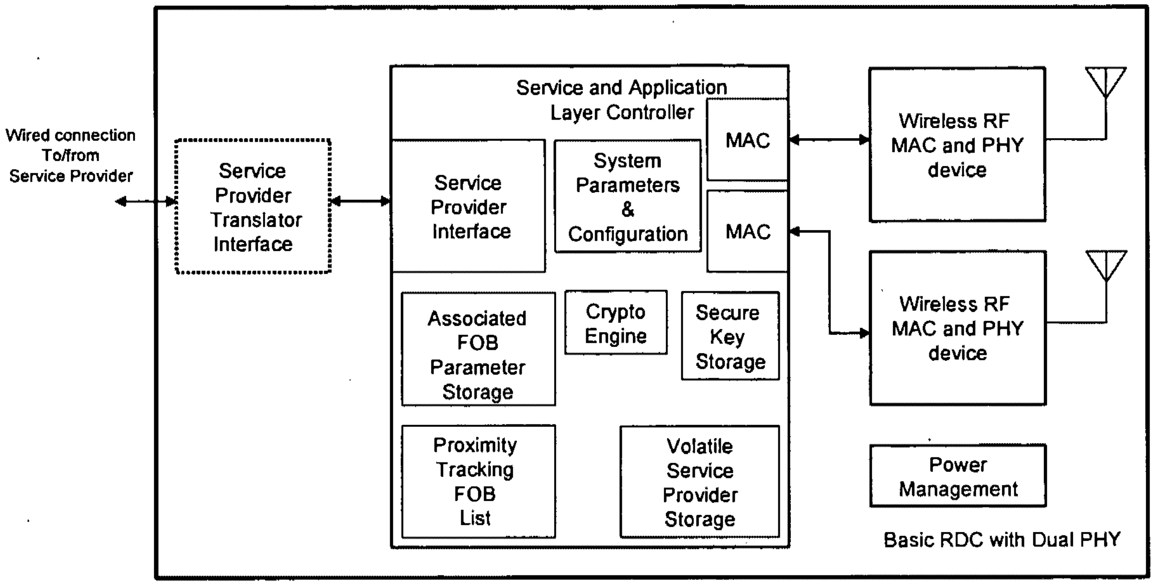

Wired connection
To/from
Service Provider
Service Provider Translator Interface
Service and Application Layer Controller
MAC
MAC
Service Provider Interface
System Parameters & Configuration
Associated FOB Parameter Storage
Crypto Engine
Secure Key Storage
Proximity Tracking FOB List
Volatile Service Provider Storage
Wireless RF MAC and PHY device
Wireless RF MAC and PHY device
Power Management
Basic RDC with Dual PHY

Figure 10: Basic RDC Configuration with Dual PHY

As shown in Figure 10, all of the same components are present as in the Basic RDC Configuration in Figure 8, except an additional Wireless RF MAC and PHY device exists and the associated MAC functionality in the Service and Application Layer Controller.

As previously mentioned, the second Wireless RF MAC and PHY device allows for over the air synchronization. By allowing this functionality, the elimination of wire line synchronization is not required.

4.2.3 CRDC

The Coordinator RDC (CRDC) can be either a Single or Dual PHY RDC. The different in the CRDC over the RDC is the CRDC has increased RF power output, and never communicates bi-directionally with a POD.

A CRDC will be capable of communicating to another CRDC, and a CRDC will also be capable of communicating with an RDC.

CRDC to CRDC communications allows for frame synchronization and frequency planning without having to having a wired connection between the units. The same is true for CRDC to RDC communications.

Although CRDCs may communicate to each other, in some configurations, the CRDC cell boundaries don't overlap and CRDCs cannot see each other. In this case, the RDC that is between the overlapping CRDC cells will communicate to both RDC and pass frequency information between them to optimize the system.

5 Wireless Protocol

The Proxense wireless protocol is derived from a subset of the IEEE 802.15.4 protocol. The frame structure and frequency plan is derived from this specification.

5.1 Frame Structure

The frame structure is a superset of the 802.15.4 frame structure. The structure was intended to provide better battery life for the POD and minimum collisions between PODs and RDCs.

5.1.1 Timeslot

A timeslot is a period of time that information is communicated between two devices.

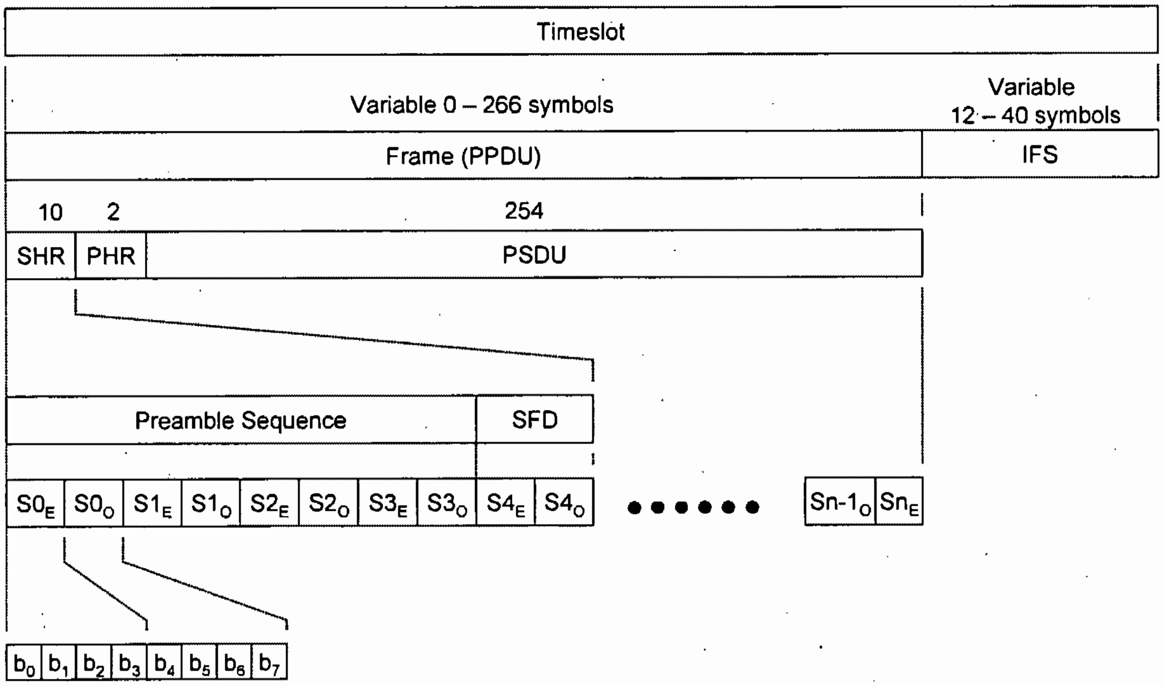


Figure 11: Timeslot / Frame structure

Figure 11 shows how the timeslot is broken down. The timeslot is broken down into a Frame or Physical Packet Data Unit (PPDU) and Inter-Frame Spacing (IFS). The PPDU contains synchronization information and carries the payload of data. The IFS allows time for a receiving end unit to process the data in the PPDU and transmitter turn-around time. Both the PPDU and IFS are variable in length and are determined by the amount of data carried in the frame.

The PPDU is then broken down into a Sync Header (SHR), a Physical Header (PHR) and a Physical Service Data Unit (PSDU). The SHR contains a preamble sequence and Start-of Frame Delimiter (SFD) which allows a receiving device to acquire the RF signal and synchronize to the frame.
The PSDU is then used to carry both 802.15.4 MAC and user data information.
The PSDU is of variable length which is determined by the type of MAC and data information being carried.

The frame is then further broken down into symbols, which in turn are broken down into bits. Each symbol contains 4 bits which are sent least significant bit (lsb) to most significant bit (msb) at the base band level.

5.1.2 Superframe

A superframe is comprised of multiple timeslots (frames with IFS). The superframe is used in a beacon enable synchronous network where PODs can find an RDC and/or CRDC fixed device. The superframe allows a POD to efficiently determine if an RDC is present on any given frequency.

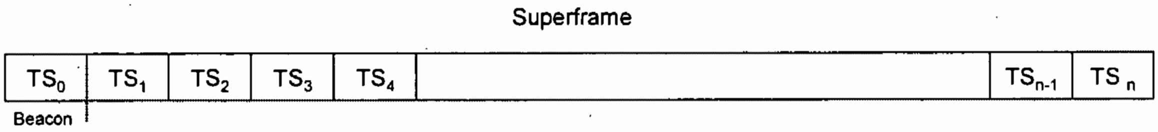


Figure 12: Superframe structure

The superframe is configured such that Timeslot 0 is the beacon timeslot. Each superframe that is transmitted always starts with a beacon timeslot. Each timeslot is equally spaced so that a POD and RDC can communicate.
The superframe is of variable length with the resolution to a timeslot.
The initial number of timeslots within a superframe is 16.

5.1.3 C-Superframe

A C-Superframe is a Coordinator Superframe and is comprised of multiple Superframes. The C-superframe provides several advantages over the superframe. The C-Superframe provides better battery management for the POD as well as provides distributed superframe and timeslots in a high density networking environment.

The C-Superframe is only generated by a coordinator RDC in the system.
As shown in Figure 13, the C-Superframe has multiple superframes. Since each superframe has a beacon, multiple beacons are transmitted per C-Superframe. This allows the POD to quickly determine if it is within a system. The C-Superframe also numbers

the superframes, so both the RDC and POD realize their position within the framing structure.

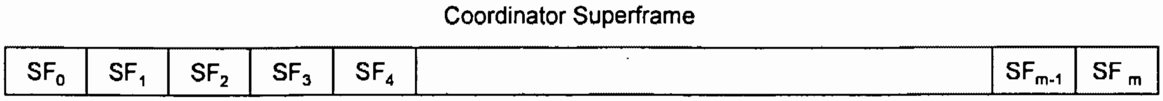


Figure 13: C-Superframe structure

In a later section, the relevance of an RDC and POD will be explained.

5.1.4 Overall Framing Structure

As a brief overview of the framing structure, Figure 14 shows how the framing is related in the Proxense protocol.

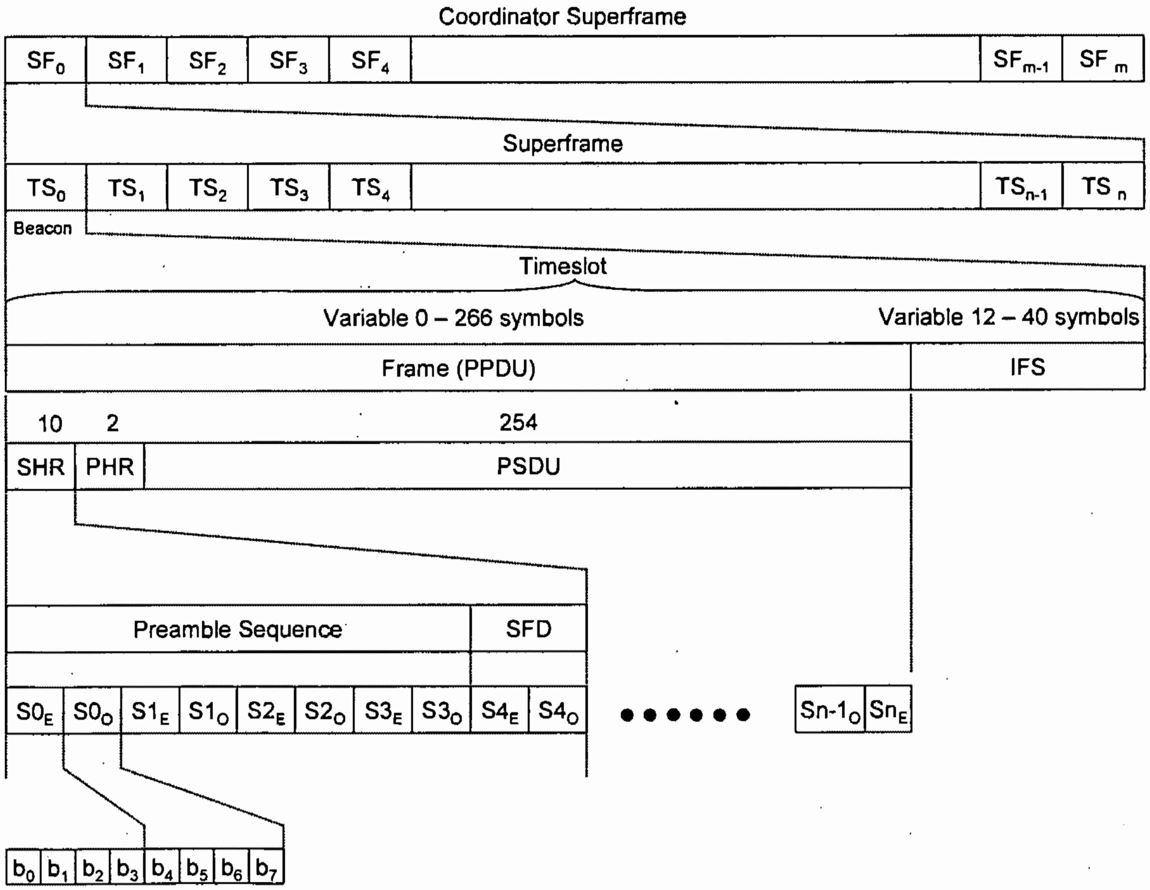


Figure 14: Overall Framing structure

5.2 Protocol Dependencies

In order for the Proxense system to operate in an efficient manner there are several protocol dependencies required. Some of these dependencies are inherent in the 802.15.4 protocol, and some are a superset created specifically for the Proxense system.

5.2.1 Beacons

As previous mentioned in section 5.1.2, the beacon is sent on every superframe. The beacon is used to alert PODs (and RDCs when a CRDC is present) of system information and timing of the framing structure employed.

In a typical single cell configuration where only one RDC is present, the beacon is transmitted in timeslot 0 of a superframe boundary. By transmitting the beacon periodically, the POD can wake up and find the beacon within a short period of time and realize that it is within a Proxense network. This configuration can be implemented with the standard IEEE 802.15.4 protocol.

In a typical unsynchronized multi-cell configuration where multiple RDCs are geographically located near each other, but no synchronization between RDCs are implemented. The POD can still wake up, detect the presence of the RDCs and synchronize and communicate to each RDC due to the presence of the beacon on each RDC. This configuration can also be implemented with the standard IEEE 802.15.4 protocol.

In a high density area where multiple RDCs are present, a CRDC will most likely be present. In this configuration, only the CRDC transmits the beacon, and all RDCs and PODs align to the CRDC beacon. In this configuration, the CRDC will send a beacon on each superframe, but will also send C-superframe and other configuration to the RDCs and PODs. This configuration will require a superset of the IEEE 802.15.4. protocol

5.2.1.1 RDC Beacon

As shown in Figure 15, in a single cell configuration the beacon is periodically output based on a specific number of timeslots.

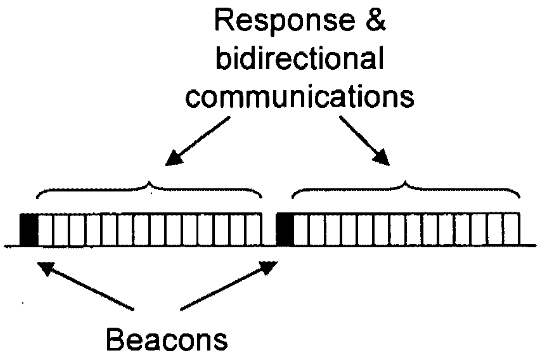


Figure 15: Beacons in a single cell configuration

It is assumed that the beacon will follow the recommendations of IEEE 802.15.4 with additional data attached indicating it is a Proxense RDC.

After a beacon is transmitted, a POD may immediately respond provided it follows the rules of Carrier Sense Multiple Access – Collision Avoidance (CSMA-CA). If the POD finds the channel is busy in the current timeslot, the POD will back-off and attempt again to access the RDC in another timeslot following the same rules. In the case the POD is not able to communicate with the RDC, it will wait for the next beacon and attempt again.

At the end of each superframe and additional idle period exists, which allows tolerance in the over-the-air protocol.

5.2.1.1.1 Typical Single Cell configuration and handshake

A typical Personal Computer application example is shown in Figure 16 below.

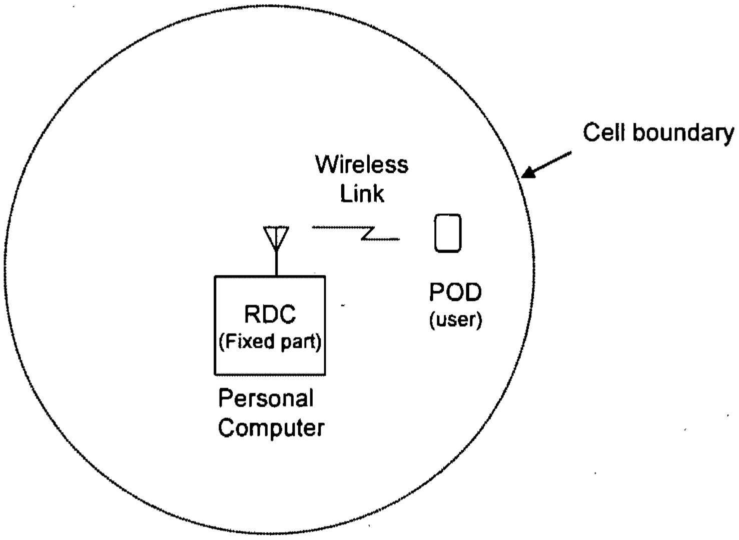


Figure 16: Personal Computer Application system architecture

Three basic components exist in the Personal Computer application. The first component is the Personal Computer with an operating system and Proxense security driver which interfaces to the RDC. For this example, the Proxense driver and RDC allow a user which has a POD to access the Personal Computer. Second is the RDC which is the gateway for the POD and performs the authorization. And third, is the POD which contains the necessary security information and must be within reasonable proximity to communicate with the RDC.

An example handshake of the POD to an RDC incorporated into a Personal computer is shown in Figure 17 below. The RDC within the Personal Computer outputs a periodic beacon based on timeslot 0 of a Superframe. Eventually a user walks within the geographical area of the Personal Computer and the POD detects the beacon from the RDC.

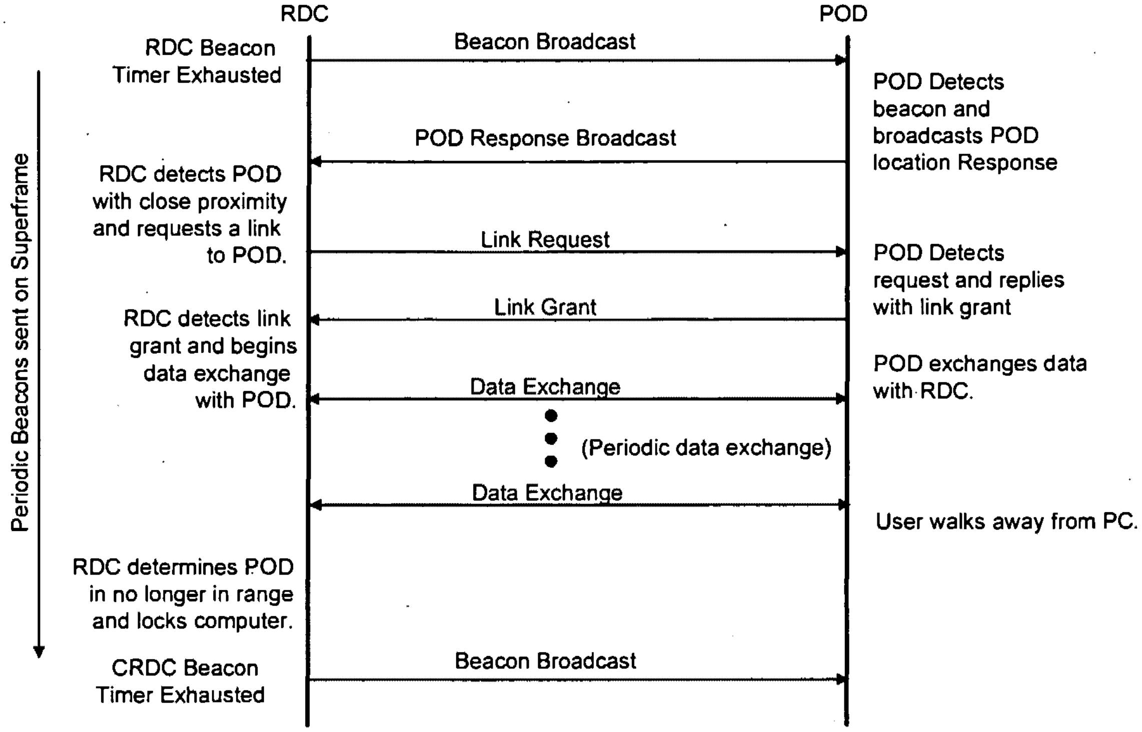

RDC
POD
RDC Beacon Timer Exhausted
Beacon Broadcast
POD Detects beacon and broadcasts POD location Response
POD Response Broadcast
RDC detects POD with close proximity and requests a link to POD.
Link Request
POD Detects request and replies with link grant
Link Grant
RDC detects link grant and begins data exchange with POD.
Data Exchange
POD exchanges data with RDC.
(Periodic data exchange)
Data Exchange
User walks away from PC.
RDC determines POD in no longer in range and locks computer.
CRDC Beacon Timer Exhausted
Beacon Broadcast
Periodic Beacons sent on Superframe

Figure 17: Personal Computer Application handshake example

The POD determines if it is registered to this RDC and if so, responds with a POD location response.

The RDC then detects the PODs response and performs a link request to the POD. The POD then accepts the request by replying with a link grant and the two devices are now in data exchange mode.

In data exchange mode the 2 devices transfer specific security information which result in the RDC enabling access to the Personal Computer through the Proxense security driver attached to the operating system.

Periodically data is exchanged between the RDC and the POD to guarantee the user is still within close proximity of the Personal Computer. As long as data exchange continues on a periodic basis, the Personal Computer remains unlocked and the user can continue to interact with their applications on the computer.

After some amount of time, the user walks away from the Personal Computer causing the data exchange to cease, and the Proxense driver locks the Personal Computer to prevent unauthorized use.

Regardless of data exchange, the RDC continues to transmit periodic beacons to guarantee other Proxense devices can gain access to the Personal Computer or other applications within the personal computer.

5.2.1.2 Coordinator Beacon (C-Beacon) Configuration

The coordinator beacon is generated by a CRDC or RDC acting like a CRDC. The CRDC covers a large geographic area covering all RDCs and PODs within that area. The C-Beacon is a standard beacon sent in the first timeslot of each Superframe as shown in Figure 18.

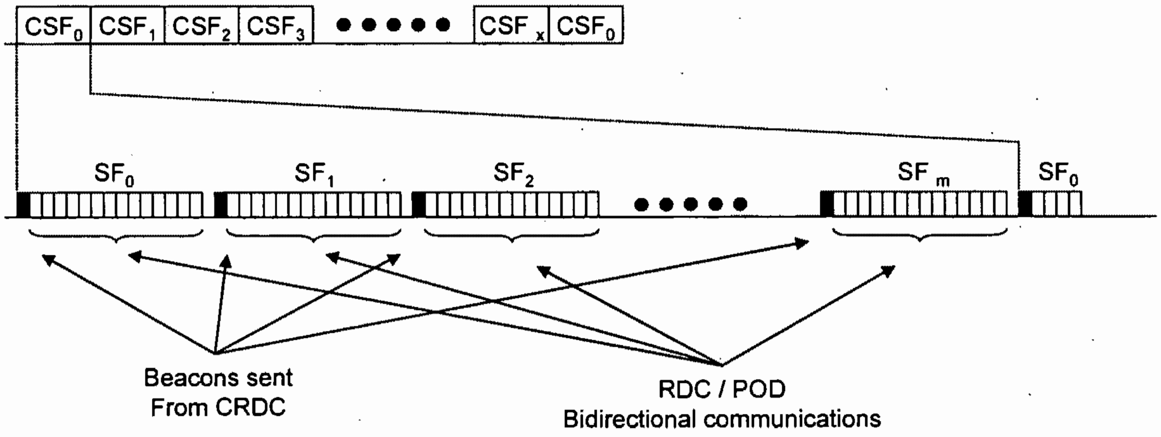


Figure 18: C-Beacon structure

There are several distinct differences in the C-Beacon over the 802.15.4 standard beacon.

- The standard beacon carries a field indicating the beacon is a C-Beacon.
- The C-Beacon in normal operation is a unidirectional transmission from the CRDC.
- The beacon also contains other C-Beacon related information:
    - Number of slots in a Superframe
    - Number of Superframes in a C-Superframe
    - The channels on which adjacent CRDCs operate.
    - Current Superframe number
    - Current C-Superframe number
    - Site ID
    - CRDC ID
    - POD Superframe mask.
    - POD Timeslot mask.
- How RDCs and PODs are coordinated in the framing structure.

While beacons are transmitted from the CRDC on timeslot 0 of each Superframe, remaining timeslots of a Superframe are left open for unsynchronized communications between PODs and RDCs.

The term unsynchronized is used for communications between the PODs and the RDCs since the RDC and POD share a common CRDC beacon, but the POD does not synchronized directly to an RDC beacon.

In this manner, the POD and RDC look like a peer-to-peer network.

5.2.1.2.1 Coordinator Beacon configuration fields

The C-Beacon information that was previously mentioned is the configuration fields that allow the system to operate efficiently when using a CRDC. In the case of a large scale system, the service provider of the system must have knowledge of RDC coverage relative to the CRDC. The fields defined below will require a header field, but none have been assigned as of this time.

The following subsections provide details of these fields. A further description and implementation example follow in section 5.2.1.2.2 below.

5.2.1.2.1.1 Superframe_Len

The Superframe_Len is governed by the IEEE 802.15.4 specification section 7.2.2.1.2. The number of slots may be from 2^1 to 2^{14}. The number of slots in a Superframe defines the repetition rate for the beacon.

5.2.1.2.1.2 C-Superframe_Len

The C-Superframe_Len is part of the Proxense system definition. This defines a higher layer counter used for extended power savings in the POD.
The C-Superframe_Len also defines the number of beacons within a Superframe.

If the Superframe is configured to not have a beacon, then this field is ignored.

Name	Type	Valid Range	Description
C-Superframe_Len	Integer	0 to 15	Defines the number of Superframes in a C-Superframe. Number of Superframes is defined as $2^{\text{C-Superframe\_Len}}$

Table 1: C-Superframe_Len

5.2.1.2.1.3 CRDC_Chan_Flags

The CRDC_Chan_Flags field indicates to the POD which channels are used by adjacent CRDCs.

Name	Type	Bit	Description
CRDC_Chan_Flags	Binary		When any bit in this field is set to a 1, an adjacent CRDC is transmitting on that frequency.
	Binary	0	1 = Channel 0 available 0 = Channel 0 not available
	Binary	1	1 = Channel 1 available 0 = Channel 1 not available
	Binary	2	1 = Channel 2 available 0 = Channel 2 not available
	Binary	3	1 = Channel 3 available 0 = Channel 3 not available
	Binary	4	1 = Channel 4 available 0 = Channel 4 not available
	Binary	5	1 = Channel 5 available 0 = Channel 5 not available
	Binary	6	1 = Channel 6 available 0 = Channel 6 not available
	Binary	7	1 = Channel 7 available 0 = Channel 7 not available
	Binary	8	1 = Channel 8 available 0 = Channel 8 not available
	Binary	9	1 = Channel 9 available 0 = Channel 9 not available
	Binary	10	1 = Channel 10 available 0 = Channel 10 not available
	Binary	11	1 = Channel 11 available 0 = Channel 11 not available
	Binary	12	1 = Channel 12 available 0 = Channel 12 not available
	Binary	13	1 = Channel 13 available 0 = Channel 13 not available
	Binary	14	1 = Channel 14 available 0 = Channel 14 not available
	Binary	15	1 = Channel 15 available 0 = Channel 15 not available

Table 2: CRDC_Chan_Flags

5.2.1.2.1.4 Superframe_Cnt

The Superframe_Cnt is part of the Proxense system definition. This field defines the current Superframe (or beacon) count within the C-Superframe.

If the Superframe is configured to not have a beacon, then this field is not transmitted.

Name	Type	Valid Range	Description
Superframe_Cnt	Integer	0 to 65535	Defines the current Superframe count.

Table 3: Superframe_Cnt

Note: Although this field is currently defined as a Proxense protocol specific flag, the IEEE 802.15.4 also defines a Beacon Sequence Number (BSN) which is an 8 bit value that can only range from 0 to 255. After further analysis, this value may become the 8 least significant bits of the Superframe Count.

5.2.1.2.1.5 C-Superframe_Cnt

The C-Superframe_Cnt is part of the Proxense system definition. This field defines the current C-Superframe count.

If the Superframe is configured to not have a beacon, then this field is not transmitted.

Name	Type	Valid Range	Description
C-Superframe_Cnt	Integer	0 to 65535	Defines the current C-Superframe count.

Table 4: C-Superframe_Cnt

5.2.1.2.1.6 POD_SF_TS_Msk

The POD_SF_TS_Msk field defines the bits of the Superframe count and the timeslot count to use for POD Superframe and Timeslot sequencing while in tracking mode.

If the Superframe is configured to not have a beacon, then this field is not transmitted.

Name	Type	Valid Range	Description
POD_SF_TS_Msk			Defines which bits are to be used for determining POD superframes and timeslots to communicate with an RDC during location tracking.
Superframe Mask	Binary	0000000000000 to 1111111111111	Defines the Superframe mask 1 = enable bit 0 = mask bit
Timeslot Mask	Binary	000 to 111	Defines the Timeslot mask 1 = enable bit 0 = mask bit

Table 5: POD_SF_TS_Msk

The POD_SF_TS_Msk value is used in conjunction with a portion of the service provider unique POD Identification value and is used to determine the exact superframe and timeslot the POD is permitted to transmit a location identifier back to the RDCs.

The necessary logic and variables required to perform this operation is illustrated in Figure 19 below.

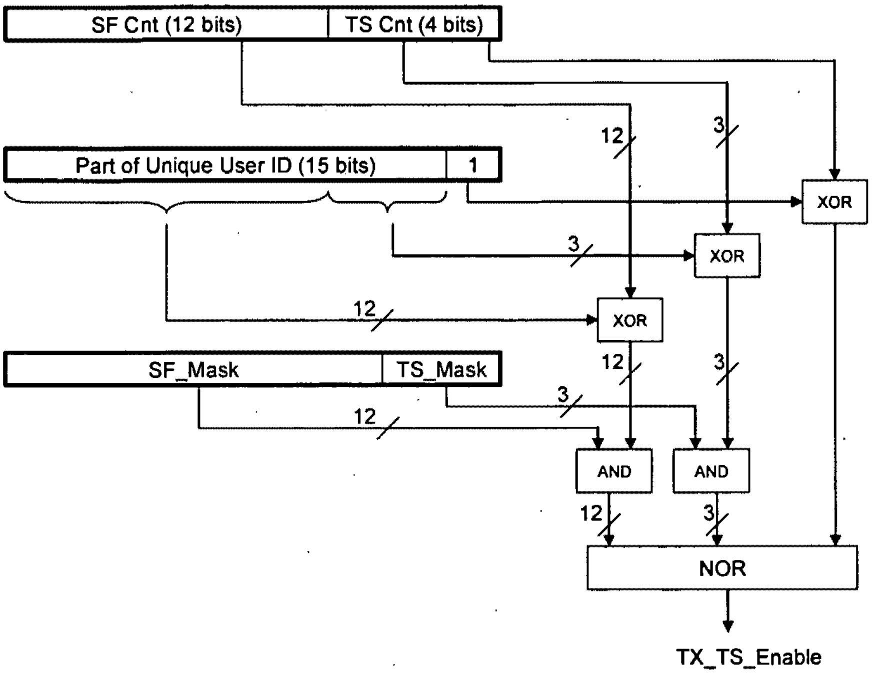


Figure 19: POD Transmit Timeslot Enable Function

5.2.1.2.1.7 Site_ID

The Site_ID is part of the Proxense system definition. Each CRDC will transmit the Site_ID to all PODs and RDCs. The Site_ID allows a POD to determine if it can access the current site or if it needs to request permissions to access the sites network.

Name	Type	Valid Range	Description
Site_ID	Integer	0 to 65535	Defines the current sites ID.

Table 6: Site_ID

5.2.1.2.1.8 CRDC_ID

The CRDC_ID is part of the Proxense system definition. Each CRDC will transmit a specific CRDC_ID to all RDCs and PODs within its cell boundary. The CRDC_ID may be used for geographical reference, in the case a CRDC must relocate channels.

Name	Type	Valid Range	Description
CRDC_ID	Integer	0 to 65535	Defines the current CRDC ID.

Table 7: CRDC_ID

5.2.1.2.2 Typical use of a C-Beacon

A typical location tracking example follows with a basic system configuration as shown in Figure 20 below.

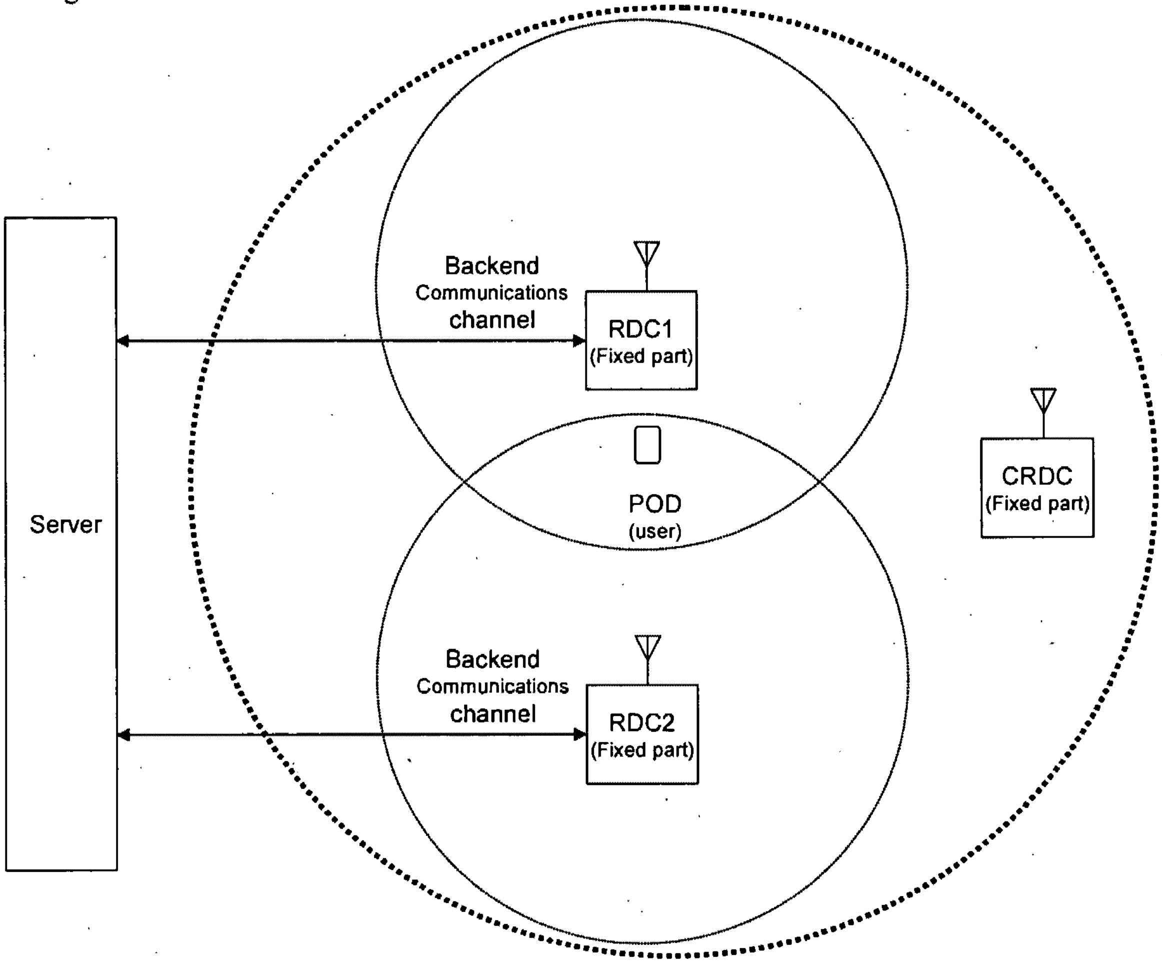


Figure 20: Basic location tracking system configuration

There are 4 different types of devices in the system. First, there is a CRDC that provides system information and synchronization for RDCs and PODs. Second are RDCs which for this example listen for PODs and report the status of each POD found within its cell boundary. Third is a POD which for this example is mobile and is moving around. And forth is the server, which is the backend computer that acquires information from the RDCs and may provide a graphical representation to personnel via a computer monitor.

For this example battery conservation is not considered and all CRDCs, RDCs, and PODs are located on the same frequency. The purpose of this example is to show how location tracking of a POD is possible and the basic handshake between different parts of the system.

A typical handshake example of POD location tracking in a CRDC configuration is shown in Figure 21.

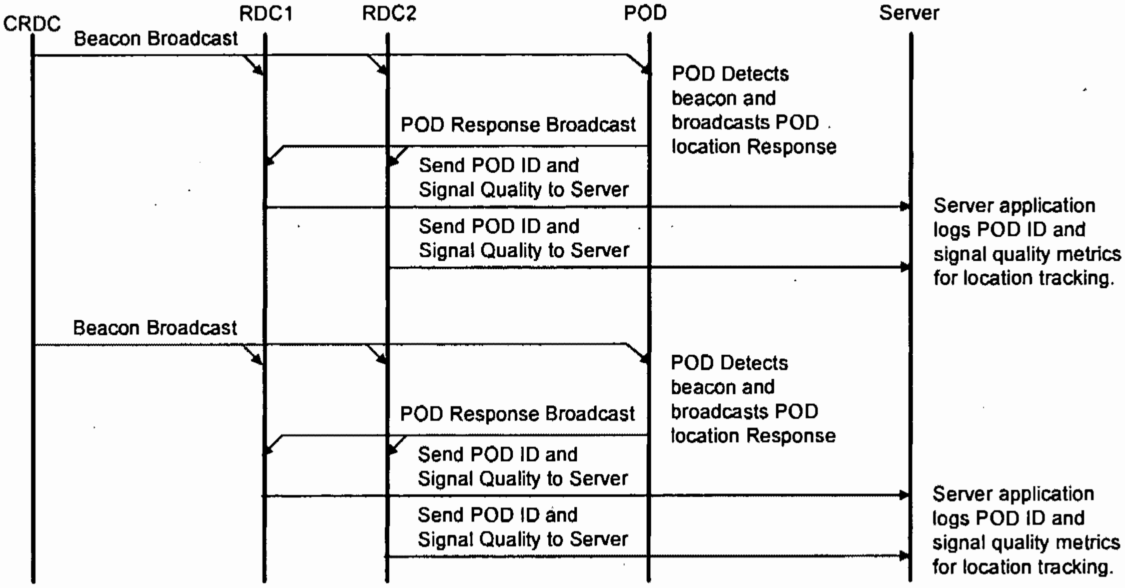

CRDC
RDC1
RDC2
POD
Server
Beacon Broadcast
POD Detects beacon and broadcasts POD location Response
POD Response Broadcast
Send POD ID and Signal Quality to Server
Send POD ID and Signal Quality to Server
Server application logs POD ID and signal quality metrics for location tracking.
Beacon Broadcast
POD Detects beacon and broadcasts POD location Response
POD Response Broadcast
Send POD ID and Signal Quality to Server
Send POD ID and Signal Quality to Server
Server application logs POD ID and signal quality metrics for location tracking.

Figure 21: Basic CRDC location tracking handshake

The CRDC periodically broadcasts a Beacon in timeslot 0 of each Superframe. All Proxense devices within the CRDC cell boundaries receive the beacon.
After the POD receives the beacon and determines that the beacon is from a system that it registered to, the POD broadcasts a POD location response which is received by all RDCs in the local geographical area.
Both RDC1 and RDC2 receive the response, log the POD ID, the signal quality metrics, and timestamps the information. The packet of information is then sent to the server where the server processes the data from each RDC and performs a location estimation which is then presented to the computer operator.

At the beginning of the next Superframe the beacon is again transmitted and the process is repeated until the POD can no longer be heard by being out of range.

5.2.1.2.3 POD and RDC coordination in a CRDC cell

Both PODs and RDCs are coordinated within a CRDC cell boundary. Since RDCs are stationary devices and may only occasionally be relocated, the RDCs are first coordinated by manually configuring both timeslots and frequencies they operate on.

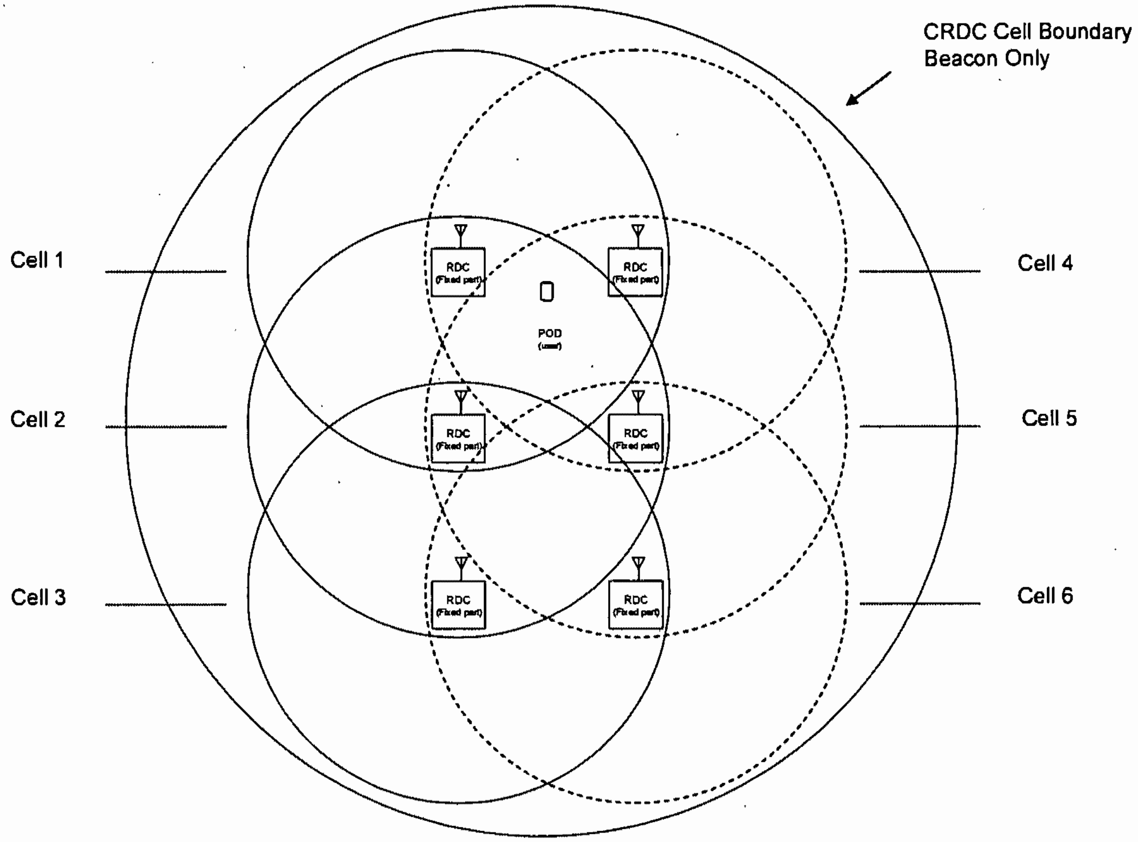


Figure 22: RDCs located in a CRDC cell boundary

As shown in Figure 22, one CRDC cell and 6 smaller RDC cells exist. The CRDC cell provides ubiquitous coverage to all of the RDC cells. Each RDC cell overlaps its adjacent RDCs in a manner that a high rate of collisions will occur if all of the RDCs attempt to talk to a POD on the same channel. It could easily be envisioned that all RDCs could be on different frequencies, but then the POD would be required to access each frequency for some duration resulting in reduced battery life.

To eliminate interference between RDCs and provide the POD with an efficient means to interact with the system, the system in Figure 22 is employed. Although this is a small network, it can be easily envisioned how this will work on a much larger scale.

To optimize the system for POD battery conservation, the each RDC contains a dual RF physical interface. The primary interface is for monitoring the C-Beacon, the POD located in close proximity, and to signal the POD to switch to another channel for further communications with that particular RDC.

For this example, the CRDC will transmit C-Beacons where all RDCs and PODs will gain timing synchronization.

Based on the configuration provided, the C-Beacon fields defined in section 5.2.1.2.1 are now configured as follows:

- Superframe_Len = 4 (2^4 = 16 timeslots)
- C-Superframe_Len = 4 (2^4 = 16 Superframes)
- CRDC_Chan_Flags = b0000000000000010 (msb to lsb – CRDC channels – only this one)
- POD_SF_TS_Msk = b0000000000011111 (mask all but 2 lsb's of the Superframe Count and don't mask any timeslot bits)
- Site_ID = 0x1234 (arbitrary site identification)
- CRDC_ID = 0x0001 (arbitrary CRDC_ID)

One other piece of information that is inherent to the POD is a unique Service Provider POD ID. The unique Service Provider POD ID located in the POD is compared with the Superframe and Timeslot Count prior to applying the mask, but does not affect the Superframe and Timeslot counts from a time reference standpoint. For this example, the unique Service Providers POD ID for this POD will be 0x0003.

Using the above values for the C-Beacon creates the following system attributes.
The Superframe is 16 timeslots long, so once out of every 16 timeslots a C-Beacon is created allowing the POD to determine if a Proxense system with the correct System exists in worse case 83 milliseconds (or 17 timeslots in the case the first part of a beacon is missed when attempting acquisition).
The C-Superframe length is set to 16 for this example.
The CRDC_Chan_Flags indicate to the POD the number of CRDC channels available in the system.
The POD_SF_TS_Msk indicates which bits to logically AND with the superframe and timeslot count to determine which slots to respond on. In this case the POD_SF_TS_Msk is a hex value of 0x001F which is ANDed with the superframe and timeslot count resulting in only one transmit timeslot.
The Site_ID and CRDC ID's are arbitrary values and will be up to the service provider to select unique identification values.

Using the above system configuration information and having a POD with a unique Service Provider ID of 0x0003, Figure 23 depicts how the POD will operate in a CRDC framing structure.

As shown in the figure, the C-Superframe_Len was set to 16, so the Superframe count counts from 0 to 15 and then starts over at 0. Each Superframe then contains 16 timeslots of which the first timeslot is timeslot 0 and always contains the beacon. The

Superframe_Len was also set to 16, so there are 16 timeslots for each superframe. Again, the timeslots are numbered from 0 to 15, and restart at 0 for each Superframe.

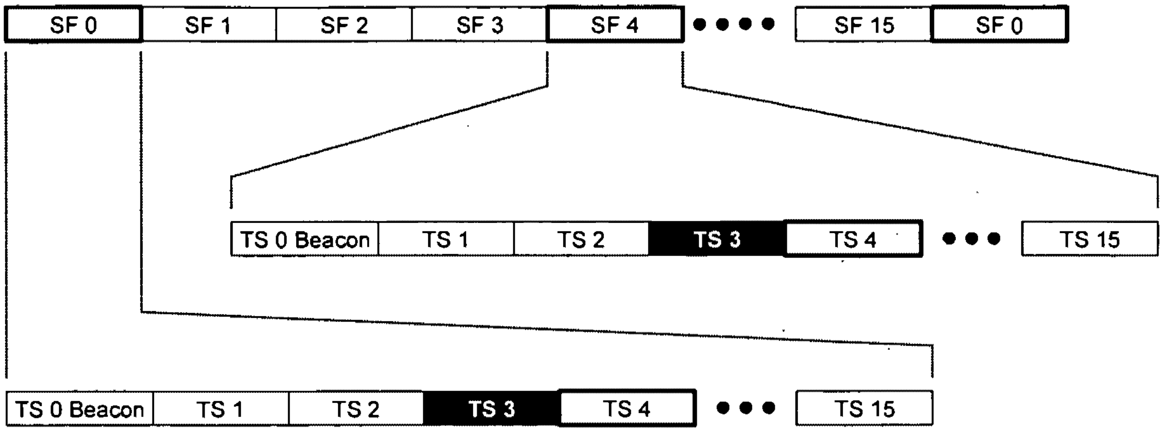


Figure 23: CRDC Framing and POD Timeslot response example

Based on the parameters set by the system and the unique Service Provider POD ID, the POD will periodically transmit a POD location response in timeslot 3 of each superframe on a modulo 4 basis. This causes the POD to respond in timeslot 3 of Superframes 0, 4, 8, and 12 of a C-Superframe. It should be noted that the POD follows the CSMA-CA rules and if the POD cannot respond in it's timeslot, it must wait for its next designated superframe and timeslot to respond.

If an RDC wanted to begin communications to a POD, the RDC would immediately respond on the next even timeslot, which in this case is timeslot 4. Any RDC could respond, but RDCs must use the CSMA-CA rule prior to responding to the PODs transmission.
If an RDC begins communications to a POD, only the following timeslot can be used to instruct the POD to go to another channel, where bidirectional communications can commence.

An active superframe occurs when the unmasked bits in the superframe count equal the corresponding unmasked bits in the unique Service Providers POD ID. For this example, the superframe mask is a value of 0x003 and the unique Service Providers POD ID is 0x0003.

With this information the following calculation occurs:

```
      b000000000000    superframe count[15:4]
xor   b000000000000    unique Service Provider POD ID[14:3]
      b000000000000    result of xor function
and   b000000000111    Superframe Mask[11:0]
      b000000000000    result of AND function
        nor all bits
              1        result is true
```

As shown above, a portion of the superframe count is exclusive-ored with a portion of the unique Service Provider POD ID. The result of the exclusive or is all 0's. Then the superframe mask is ANDed with the result of the exclusive-or function. The AND operation also results in all 0's. The result of the AND function is then compared to zero by NORing all of the bits together and result in a 1 or true output, indicating the bits compared between the superframe count and the unique Service Provider POD ID are a match.

An active timeslot occurs when the unmasked bits in the 3 most significant positions of the timeslot count equal the unmasked bits in the unique Service Providers POD ID's least 3 significant bits and the timeslot count lsb is a 1(POD only transmits on odd frames). For this example, the timeslot mask is a value of 0x7 and the unique Service Providers POD ID's 3 lsb's are 0x3.

With this information the following calculation occurs:

	b011	timeslot count[3:1]
xor	b011	unique Service Provider POD ID[2:0]
	b000	result of xor function
and	b111	Timeslot Mask
	b000	result of AND function
	nor all bits	
	1	result is true

As shown above, the timeslot count is exclusive-ored with a portion of the unique Service Provider POD ID. The result of the exclusive or is all 0's. Then the timeslot mask is ANDed with the result of the exclusive-or function. The AND operation also results in all 0's. The result of the AND function is then compared to zero by NORing all of the bits together and result in a 1 or true output, indicating the bits compared between the timeslot count and the unique Service Provider POD ID are a match.

The last portion of the calculation that needs to be completed (as previously mentioned) is to verify the last bit of the slot count is a one, indicating an odd slot.

As you will see by the above calculations, if the unmasked superframe and timeslot bits don't match the appropriate unique Service Provider POD ID, then the results will be false and no match will occur.

For this example the superframe mask was set to only unmask the 2 lsb's of the superframe count to show that it is possible to allow a POD to come up more frequently than the C-superframe count. By increasing the superframe mask to 4 bits, this example would allow the POD to only respond once per C-superframe (since the C-superframe was set to 16 and the modulo for the mask would be 2^4 or 16.

The timeslot mask was set to allow all timeslot bits to be correlated to determine the timeslot, allowing the POD to only respond once per superframe. It is possible to mask some of the timeslot bits to increase the number of times a POD can respond within a superframe.

5.2.1.2.3.1 Synchronization of a POD in a CRDC cell

A POD periodically wakes up to determine if it is within a Proxense system. Upon a periodic wake up, the POD will detect a C-Beacon indicating that a Proxense system is present along with system information.

The POD will collect the system information and determine the current superframe count of a C-Superframe. The POD will also put the parameters (POD_SF_TS_Msk, etc.) in place to start immediate battery save in the system.

Based on an approximate time, the POD will awake just prior to where it believes the next superframe is that it should communicate on, and will listen for the Beacon and begin responding with the POD location identification message.

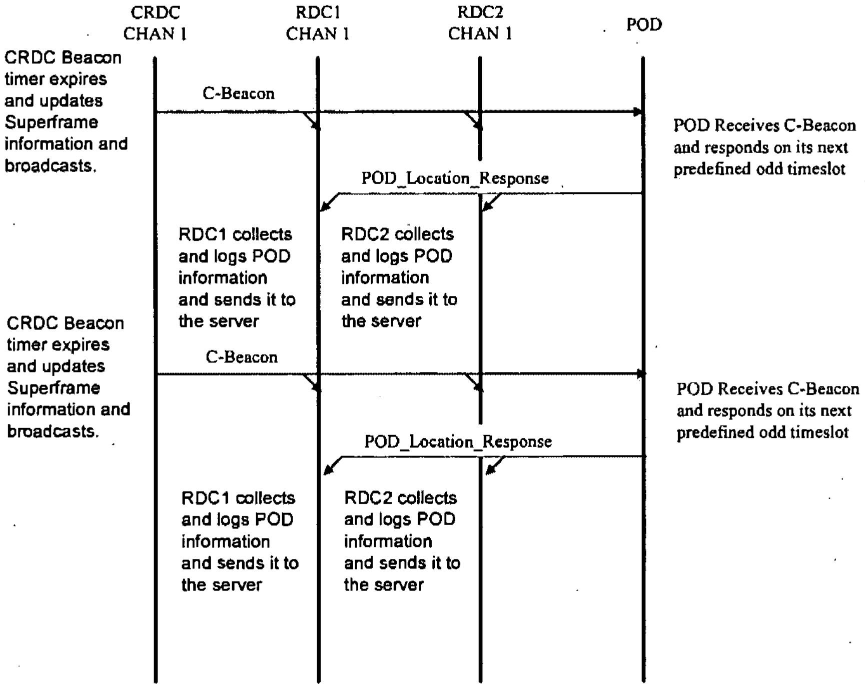

CRDC CHAN 1
RDC1 CHAN 1
RDC2 CHAN 1
POD
CRDC Beacon timer expires and updates Superframe information and broadcasts.
C-Beacon
POD Receives C-Beacon and responds on its next predefined odd timeslot
POD_Location_Response
RDC1 collects and logs POD information and sends it to the server
RDC2 collects and logs POD information and sends it to the server
CRDC Beacon timer expires and updates Superframe information and broadcasts.
C-Beacon
POD Receives C-Beacon and responds on its next predefined odd timeslot
POD_Location_Response
RDC1 collects and logs POD information and sends it to the server
RDC2 collects and logs POD information and sends it to the server

Figure 24: CRDC Beacon and POD response handshake

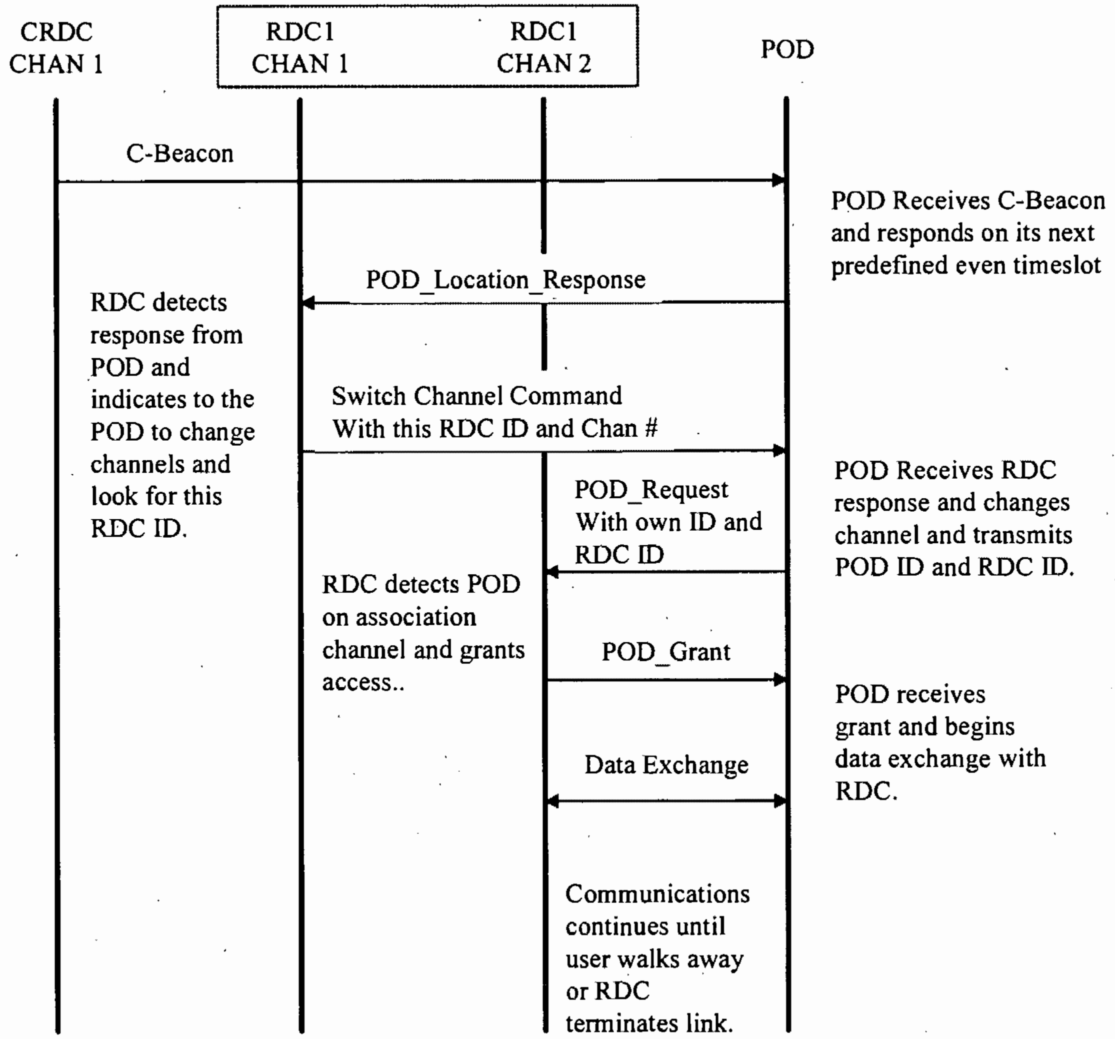
As shown in Figure 24, the CRDC updates its system information on each superframe and outputs a C-Beacon with the current information to all PODs and RDCs. The POD then waits for its predefined superframe and timeslot and responds.

This scenario continues to occur until the POD leaves the CRDC cell or an RDC responds to the POD.

5.2.1.2.3.2 POD and RDC association in the CRDC cell

As mentioned in the previous section, the CRDC continues to output a C-Beacon and the POD periodically awakes to re-align to the Superframe and respond to the C-Beacon.

If a RDC is present and wants to communicate with the POD, the RDC must respond on the even timeslot immediately available after the PODs transmission.



CRDC CHAN 1
RDC1 CHAN 1
RDC1 CHAN 2
POD
C-Beacon
POD Receives C-Beacon and responds on its next predefined even timeslot
POD_Location_Response
RDC detects response from POD and indicates to the POD to change channels and look for this RDC ID.
Switch Channel Command With this RDC ID and Chan #
POD Receives RDC response and changes channel and transmits POD ID and RDC ID.
POD_Request With own ID and RDC ID
RDC detects POD on association channel and grants access..
POD_Grant
POD receives grant and begins data exchange with RDC.
Data Exchange
Communications continues until user walks away or RDC terminates link.

Figure 25: POD / RDC association in a CRDC cell

The diagram depicted in Figure 25 above illustrates how the communications handshake between the POD and RDC occur. The diagram contains one CRDC, one RDC shown with 2 active channels (using 2 Phy's), and a POD.

In the diagram, the CRDC outputs a C-Beacon of which the RDC and POD are aligned. The POD realizes that the C-Beacons' Superframe count correlates to it's internal predefined active superframe count, and then waits for the appropriate timeslot to respond to the system with its POD location response.

When the POD responds, on the C-Beacon channel, the RDC detects the response and determines that it wants to associate with the POD. The RDC then creates a message including its own RDC ID, the PODs ID, a command to switch to channel 2, and a predicted superframe and timeslot the POD should respond on.

The POD immediately switches to channel 2 and waits for the appropriate superframe and timeslot count and transmits a link request along with its POD ID and the destination RDC ID. The destination RDC then receives the information and responds back to the POD with a link grant.

Communications can now begin between the 2 devices exchanging the appropriate information to keep the POD and RDC link.

To maintain synchronization, the RDC will define the periodic communication frequency with the POD and will periodically generate a request to the POD to exchange information.

The POD has reconfigured its wake parameters to that of the RDC, since the RDC is maintaining superframe synchronization.

Remember the RDC has a dual physical interface and it is maintaining synchronization with the CRDC on channel 1 while associating with one or more PODs on channel 2. The physical interface connected to channel 1 is providing the timing to the physical interface on channel 2.

Because the RDC has intelligence on both channels, the RDC can provide coordination of PODs that it wants to redirect to channel 2, and pods that are on channel 2. To further explain, the RDC can move the superframe and timeslot that a POD communicates to the RDC on, if another POD with the same timeslot requirements is present on channel 1 and the RDC wants to associate with it.

5.2.2 CRDC Slot and Channel coordination

The CRDC may be configured via a remote connection to a server or automatically. Using remote configuration, the server has a knowledge of RDCs located within the CRDC cell boundary and can perform optimum channel and timeslot planning.

When the CRDC is configured automatically, the CRDC must scan all channels and find the channel with the least interference. The CRDC can then begin transmitting a C-Beacon.

All RDCs located within the CRDC cell boundary will place the CRDC into its local CRDC database and complete scanning all other channels to determine if other CRDCs are present. In the case multiple CRDCs have been found, the RDC must communicate to each CRDC its findings if possible.

Each CRDC will then coordinate through that RDC to setup channels and timeslots to prevent interference between CRDCs. In most cases the CRDC will select another channel and will disregard the timeslot information since CRDCs are not required to be timing coordinated.

Any RDC that detects more than one CRDC will select the CRDC with the best signal quality.

5.3 Proxense Protocol Operation

The subsection will describe the protocol operation in a single cell and in a CRDC configuration.

5.3.1 Additional Proxense Protocol Fields

There are additional protocol fields required to allow interoperability between single cell and coordinated cell configurations.
The additional protocol fields provide information to RDCs and PODs that are located in near proximity to each other, or within a CRDC cell.
The fields defined below will require a header field, but none have been assigned as of this time.

5.3.1.1 Proxense Network Format

The Proxense network format field provides information to RDCs and PODs related to the specific configuration the single cell or coordinated cells are operating in.

Name	Type	Valid Range	Description
Proxense_NWK_FMT			Defines the network configuration employed for an RDC or CRDC
Network Type	Integer	0 to 7	Defines the network type employed. 0 = Single Cell 1 = Multi-Cell coordinated 2 = Multi-Cell coordinated w/ CRDC 3 – 7 = RFU
Beacon Source	Binary	0 or 1	Defines the source of the beacon. 0 = RDC 1 = CRDC
Broadcast Flag	Binary	0 or 1	Defines if this is a broadcast message. 0 = not broadcast 1 = broadcast
Timeslot Select	Integer	0 to 3	Defines how an RDC and POD utilize timeslots in a system. 0 = no timeslots assigned 1 = 802.15.4 Beacon enabled 2 = POD uses even timeslots / RDC uses odd timeslots 3 = POD uses odd timeslots / RDC uses even timeslots

Table 8: Proxense network format field

5.3.1.1.1 Network Type

The Network Type defines the cell network configuration. An RDC receiving this field will determine its operating mode based on this field.
If an RDC receives this information and determines another RDC in single cell mode is present, the RDC will know to move off of this channel to avoid collisions with the other RDC.
If the RDC receives this information and determines a Multi-Cell coordinated system is being employed, the RDC may join the Multi-Cell coordinated system.
If the RDC receives this information and determines a Multi-Cell coordinated system with a CRDC is employed, the RDC will join the CRDC network.

The POD also receives this information and adjusts its operating mode to comply with the system employed.
If the POD detects the system to be single cell, the POD will conform to more of a 802.15.4 protocol, communicating with the RDC in this manner. The POD will be aware that it is only required to communicate with a specific RDC ID. The POD must still have the capability to periodically monitor other channels for other RDCs in the local vicinity.

If the POD detects the system is Multi-Cell coordinated, the POD may receive further information indicating the other RDC frequencies in use in the coordinated network and may adjust its system operating parameters appropriately.

If the POD determines the system is Multi-Cell CRDC coordinated, the POD will adjust its operating parameters appropriately. The POD will acknowledge that a C-Beacon will be present and will broadcast a POD Location Response. The POD will also understand that an RDC with a different ID other than a CRDC ID will attempt communications with the POD.

5.3.1.1.2 Beacon Source

This field indicates to all RDCs and PODs in the general proximity of the type of Beacon being generated. This information is helpful, specifically when in a Multi-Cell CRDC system and allows the RDCs to distinguish between RDC generated beacons and CRDC generated beacons.

5.3.1.1.3 Broadcast Flag

The broadcast flag indicates to all recipients that the information being sent is intentionally being broadcast to all devices that can receive Proxense protocol information. In some cases a message that can be sent to a specific POD can also be broadcast to all PODs. This flag assists the PODs in determining how to treat the information.

5.3.1.1.4 Timeslot Select

The timeslot select field indicates to PODs and RDCs how the timeslots are configured in the system. The field further determines if an RDC and POD are to use even based, or odd based timeslots for responding.

5.3.1.2 Proxense Network Identifier

In order for an RDC or POD to determine that the network is a Proxense network, a Proxense network identifier must be present.

Name	Type	Valid Range	Description
Proxense Network Identifier	ASCII	ProXense	An 8 byte ASCII value identifying the network to be a Proxense network

Table 9: Proxense Network Identifier

5.3.2 Single Cell Standalone Operation (PC Application)

The following description is based on a PC application running in a single cell standalone configuration. The PC has a resident RDC integrated in its hardware and has a Proxense specific driver that allows access to the operating system and certain programs.

This description covers the basic initialization of the system and RDC / POD interactions that occur while associated in the system.

5.3.2.1 First time system initialization and basic link setup

Upon power being applied to the PC, the internal circuits perform an initialization function and the operating system loads and the user is provided with a graphic user interface into the operating system.

The Proxense driver also loads and the RDC device is started in a static mode with it's transceiver disabled. The RDC must be first configured to operate in the PC environment and requires some basic setup requirements.

The user must select the Proxense RDC utility and either set the RDC for auto-discover mode, or may choose to manually configure parameters related to the RDCs operation.

If the user chooses auto-discover, the RDC generates a random value for its RDC ID and its password which is presented to the user. The RDC by default also sets its network configuration to single cell.

If the PC user chooses to manually setup the RDC, the PC user may enter an RDC ID and password. The PC user (or system administrator) may also select the network topology and preferred channels the RDC will operate in. If the PC user has knowledge of the PODs public key, the user may also enter it into the setup.

The PC user then closes out their setup window and the RDC enables its transceiver.

The RDC uses the following information for configuration.

Site ID	0x0000	Generic value for a single cell RDC.
RDC ID	0x1234	Arbitrary value
C-Superframe Length	32	2.5 second superframe period for POD wakeup
Superframe Length	16	16 slots per superframe
Proxense Network Identifier	ProXense	Defines system as a Proxense system
Proxense Network Format		
Network Type	0	Single Cell
Beacon Source	0	RDC
Broadcast Flag	1	Broadcast
Timeslot Select	1	802.15.4 timeslots

Table 10: RDC Configuration information

The Site ID is set to 0, since this is a single cell RDC and no site information is required. The RDC ID is arbitrarily selected.

The C-Superframe length is set to 32 superframes indicating to the POD that it only needs to wake up once every 32 superframes in superframe 0 to exchange information with the RDC to remain associated.

The Superframe Length is set to 16, which is the standard value for a superframe in the Proxense system.

The Network Identifier allowing a POD to understand the beacon is from a Proxense technology enabled RDC.

The Proxense Network Format indicates 4 parameters.
The Network Type indicates to the POD that it is a single cell network. This will indicate to the POD no other RDC is associated with this RDC and therefore no other RDC should be attempting access on this channel.
The Beacon Source indicates to the POD that the beacon is from an RDC and not a CRDC device.
The Broadcast Flag indicates to the POD that the message is being broadcast from the RDC.
And the Timeslot Select field indicates that a POD should use 802.15.4 Beacon based handshaking with the RDC.

The RDC then scans all channels (or user preferred channels) to determine if any other 802.15.4 or Proxense devices are present or if any other interference is found.

With the preconfigured information, the RDC then begins beacon transmission on the least interfered channel with a C-Superframe count, superframe count, and the information located in Table 10: RDC Configuration information Table 10 above.

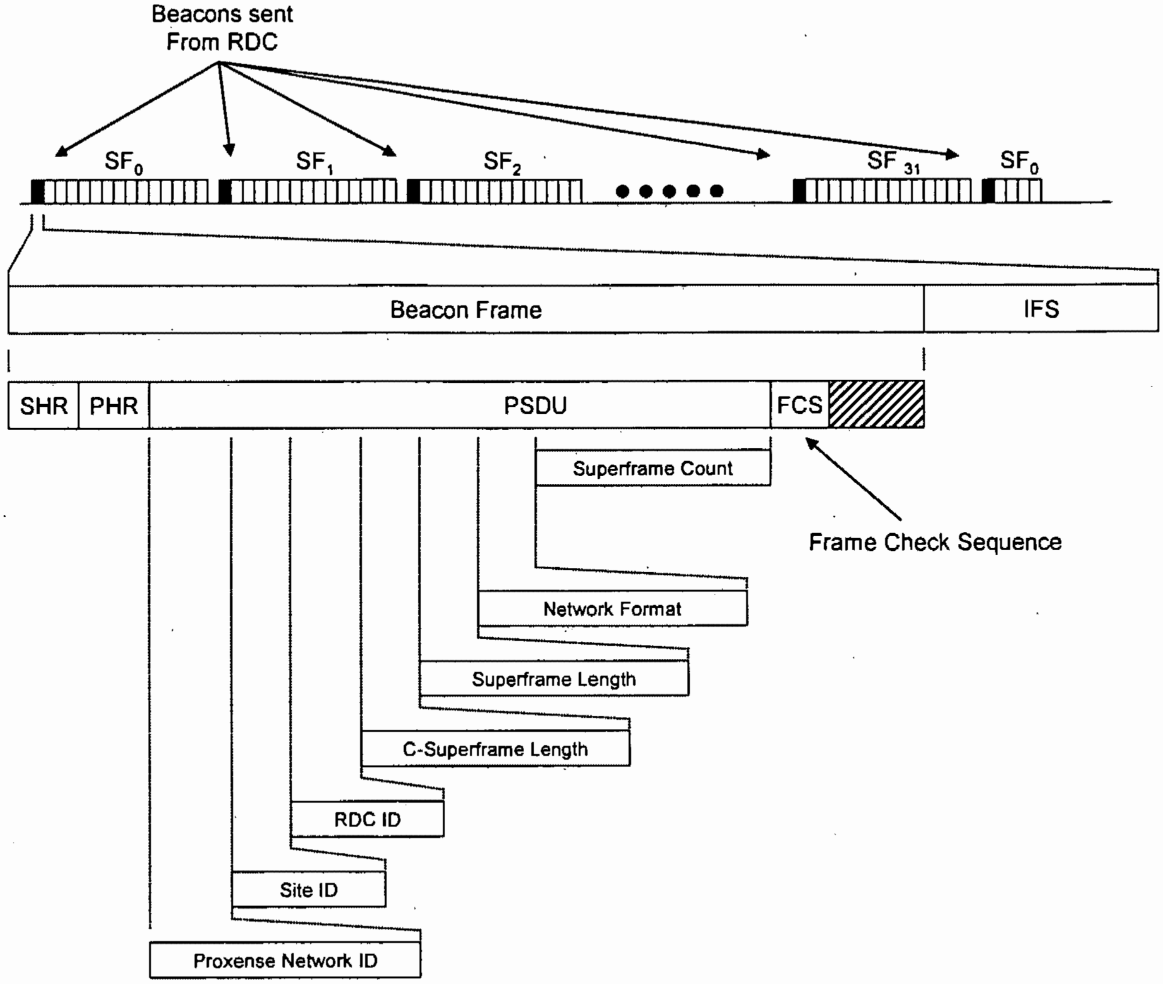


Figure 26: RDC Beacon Transmission

As shown in Figure 26, in standalone mode the RDC will continue to transmit beacons on every superframe. The information in Table 10 is transmitted along with the Superframe count in every superframe to allow the POD to configure and synchronize with the system. When the superframe count is the superframe length minus one, the superframe count will start counting from 0 for the next C-Superframe.

At the end of each Beacon transmission, a Frame Check Sequence (FCS) is appended as part of the 802.15.4 physical layer. The FCS provides protection for the data carried in the Frame. Since the beacon may not occupy the entire frame, the hashed lines are shown indicating additional idle time between the FCS and IFS.

The RDC maintain the beacon transmission until the RDC is disabled or power is removed.

At this point there are no PODs registered with the RDC, so no POD can gain access without registering and receiving authorization from the RDC.

The user acquires a POD which is not yet in the RDC cell. The POD is in battery save mode and periodically wakes up looking for a Proxense network.

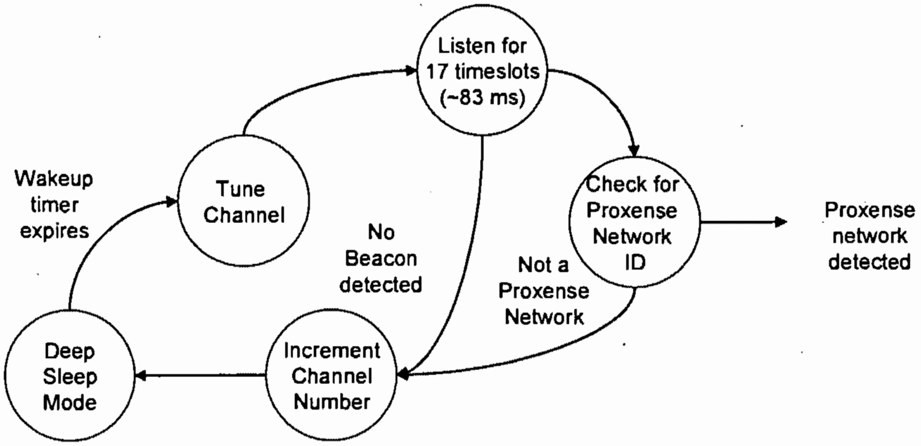


Figure 27: POD Deep Sleep diagram

Illustrated in Figure 27, the POD starts in the deep sleep state. The wakeup timer eventually expires, causing the POD to enable and tune its receiver. The POD then monitors the channel that it tuned to for a period of one 17 timeslots (one superframe plus one slot) or approximately 83 milliseconds. The 17 timeslot limit is based on a superframe of 16 timeslots, and the fact that the POD upon initial reception could miss the beginning of a beacon. The additional slot provides the overlap necessary to guarantee reception of a beacon if one is present.

If no beacon is detected, the channel number is incremented (modulo 16) and the POD resets its wakeup timer and returns to deep sleep mode. If a beacon is detected, the POD checks for a Proxense network ID and if one is not found, it again increments the channel number, resets its wakeup timer, and returns to deep sleep mode.

If the Proxense network ID is detected, the POD attempts to establish a communications link with the RDC.

At this point the POD has found a Proxense single cell network on channel 1 with a RDC ID of 0x1234, and the RDC is in broadcast mode indicating that it is attempting to gain the attention of any PODs in the local proximity.

In a first scenario, the PC application is configured to alert the PC user of PODs in the local proximity and give the PC user the option to authorize the POD for PC access. For this scenario, the PC user will decline the authorization of the POD to describe the sequence of events that will occur.

As shown in Figure 28, the RDC broadcasts its beacon with the information as described in Table 10. The POD detects the beacon in broadcast mode, determines the network configuration and RDC ID and returns a POD location response with the RDC ID and POD ID included.

The RDC detects the response from the POD and alerts the PC user that a POD with a public ID of 0x9876 (arbitrary) wants to attach and gain authorization to the PC. In the mean time the RDC immediately responds back to the POD indicating that the POD should wait for authorization. This keeps the POD responding to beacons as defined by the fields located in the beacon until the beacon is no longer present (the POD is no longer in the RDC cell), or until a response is returned by the RDC.

The user not recognizing the PODs public ID selects the "DENY" button on the Proxense utility.

The RDC continues to output its beacon. Upon the next wakeup and POD location response from the POD, the RDC detects the POD ID and within its database looks up the authorization parameters for this POD. It determines that authorization has been denied and sends an "Authorization Deny" command to the POD.

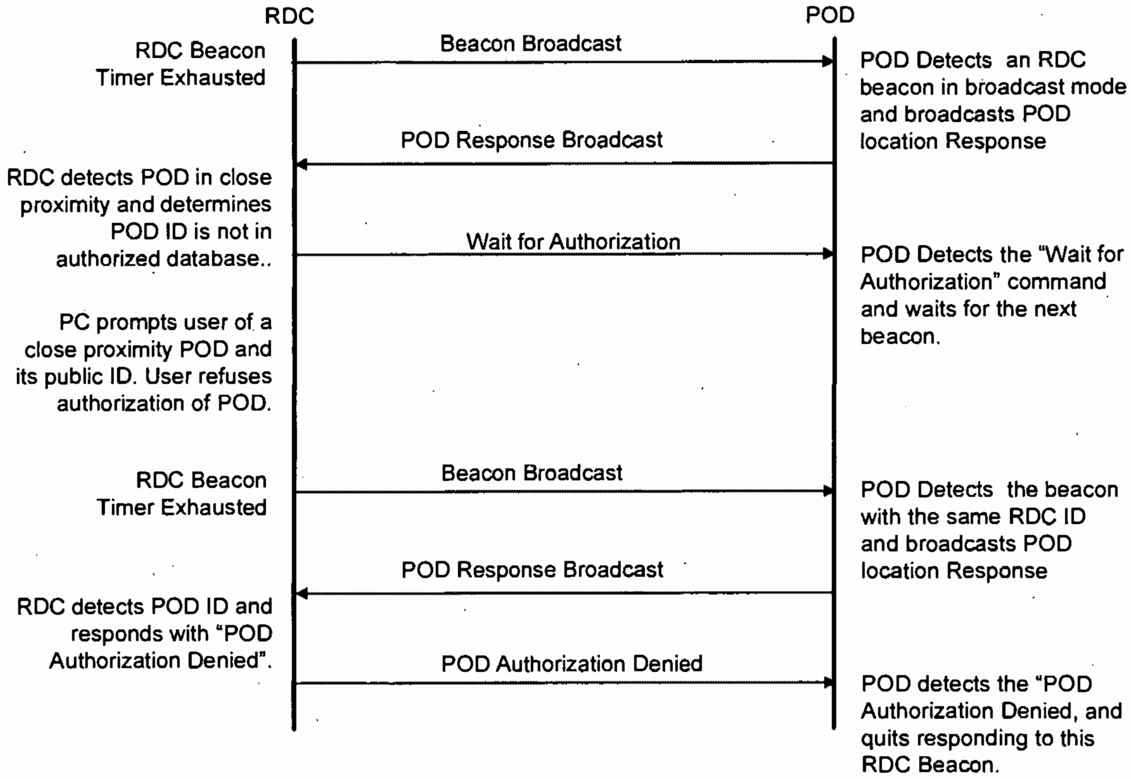

RDC
POD
RDC Beacon Timer Exhausted
Beacon Broadcast
POD Detects an RDC beacon in broadcast mode and broadcasts POD location Response
POD Response Broadcast
RDC detects POD in close proximity and determines POD ID is not in authorized database..
Wait for Authorization
POD Detects the "Wait for Authorization" command and waits for the next beacon.
PC prompts user of a close proximity POD and its public ID. User refuses authorization of POD.
RDC Beacon Timer Exhausted
Beacon Broadcast
POD Detects the beacon with the same RDC ID and broadcasts POD location Response
POD Response Broadcast
RDC detects POD ID and responds with "POD Authorization Denied".
POD Authorization Denied
POD detects the "POD Authorization Denied, and quits responding to this RDC Beacon.

Figure 28: Pod Authorization denial handshake

The POD temporarily stores the RDC ID in its local memory with a flag indicating that it shall no longer respond to this RDC ID.

The POD then goes back into battery save mode and periodically scans all channels as previously shown in Figure 27.

Since the POD is not constantly aware of its geographical location, the POD continues to monitor each channel and decode each beacon. Eventually the POD will return to the channel that the RDC is still transmitting beacons on, decode the beacon information including the RDC ID, and determine that it is not to respond.

The POD will maintain the POD ID in its local database until the beacon is no longer present during scanning, at which time the RDC ID is removed from the database.

Assuming the RDC beacon is absent for a given period of time, indicates to the POD that the POD has left the RDC cell.

Upon the next detection of that RDC beacon, the POD will once again attempt to gain access to the RDC as illustrated in Figure 28. The difference this time is the POD ID is in the RDCs local database and the RDC will deny authorization without alerting the PC user. The POD will then operate as previously described after authorization has been denied.

If the PC user decides to grant authorization, the PC user can open the Proxense utility, select the PODs public ID, and select the authorize button in a control panel.

Upon granting the POD access, one of 2 scenarios exists. First, the RDC can transmit the POD ID as part of the beacon transmission, alerting the POD that the RDC wants to communicate, and the POD then responds with a POD location response.
In the second scenario, the POD returns to the RDC cell, and after detection of the beacon (with or without the POD ID), it returns the POD location response.

In either scenario, after the RDC detects the POD location response with the PODs public ID, the RDC then issues a Link Request attempting to initiate a link between the RDC and POD.

Figure 29 illustrates the handshake between the RDC and POD for a POD authorization grant.

As shown, the RDC broadcasts the beacon on every superframe. Although not shown, the superframe counter value is also included which the POD uses for battery conservation.

The beacon is broadcast in one of 2 methods. The PC user has just authorized a POD and the beacon includes the POD public ID for a period of time, or the beacon is output without the POD public ID.

As previously mentioned if the POD is in the RDC cell and has deactivated its response, when the POD detects its ID in the beacon, it reactivates its response to the RDC and transmits the RDC ID and POD public ID in its POD location response.

If the POD has just re-entered the RDC cell and detects the RDC beacon (with or without the POD public ID), the POD will again respond with the RDC ID and its POD public ID.

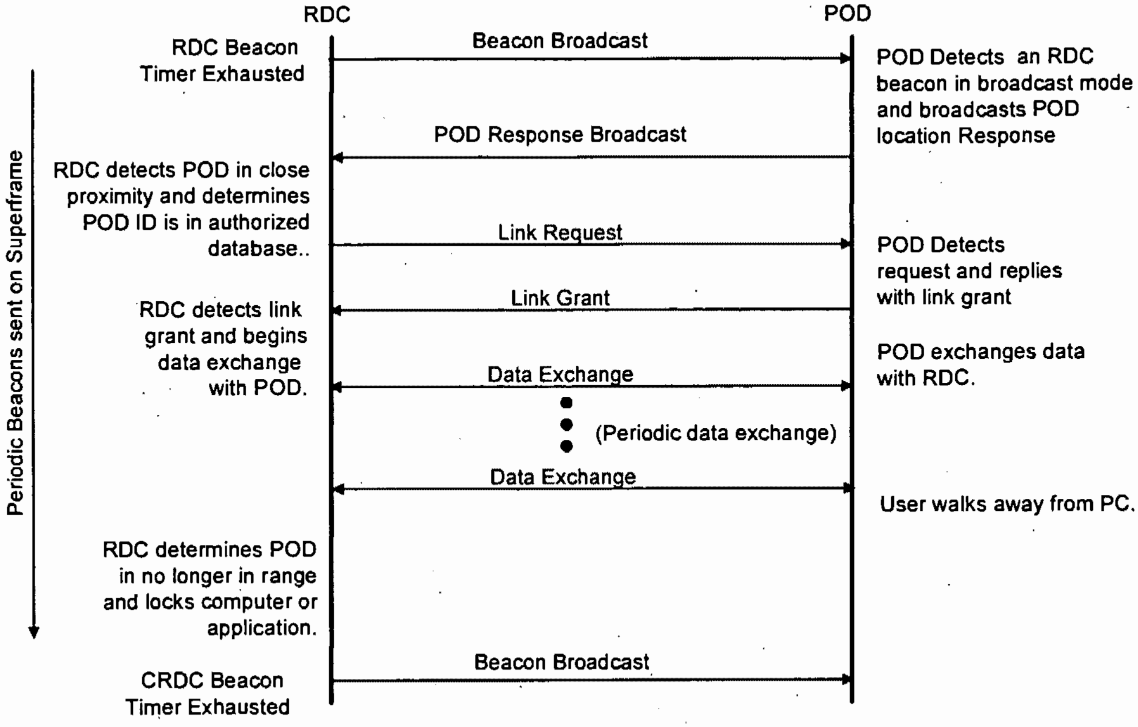

RDC
POD
RDC Beacon Timer Exhausted
Beacon Broadcast
POD Detects an RDC beacon in broadcast mode and broadcasts POD location Response
POD Response Broadcast
Periodic Beacons sent on Superframe
RDC detects POD in close proximity and determines POD ID is in authorized database..
Link Request
POD Detects request and replies with link grant
Link Grant
RDC detects link grant and begins data exchange with POD.
POD exchanges data with RDC.
Data Exchange
(Periodic data exchange)
Data Exchange
User walks away from PC.
RDC determines POD in no longer in range and locks computer or application.
Beacon Broadcast
CRDC Beacon Timer Exhausted

Figure 29: POD Authorization Grant handshake

The RDC then detects the its RDC ID and the POD public ID and immediately sends a Link Request to the POD with its RDC ID and the POD public ID indicating it wants to initiate a link with the POD.

The POD detects the request and responds with a Link Grant with both IDs included. The RDC and POD then provide data exchange on a periodic basis guaranteeing the POD remains in range of the RDC.

The periodic data exchange occurs based on the parameters previously defined in Table 10. Interleaved between the data exchange are beacons which other PODs will use to access the RDC.

Eventually, the RDC can terminate the data exchange based on inactivity or PC user interruption, or the POD will leave the RDC cell in which case the RDC realizes the POD is no longer in range. The RDC then performs the necessary security measures to secure the PC operating system, applications, or files.

It should be noted that due to radio interference issues that occur in wireless systems, the RDC and POD shall not relinquish the link based on the lack of a single data exchange. Since the RDC is not battery limited, the RDC continues to monitor all timeslots in a

superframe with the exception of those frames it transmits on. On the other hand, the POD is significantly battery limited and must intelligently choose when to receive and transmit.
In the scenario the RDC and POD lose communications during the predefined period they are attempting to communicate; both devices will have equal knowledge of this. Since the POD is battery limited, the POD will try on the next available timeslot to regain synchronization with the RDC. After a period of time in the RDC, or a predefined number of attempts by the POD, the link shall be considered lost.

5.3.2.2 Multi-access in a Proxense enabled single cell system

In the PC environment, there is a need for multiple users to have access to a single system. Once example of this is a PC user must have access to perform the duties required on a daily basis for business or personal purposes. At the same time, a system administrator or parent is required to have access to the PC for various reasons.

A Proxense enabled PC shall contain a database in which multiple POD public IDs can reside. The RDC shall maintain this database via the Proxense utility program and driver. In addition to allowing multiple users, the Proxense utility in conjunction with the RDC shall provide multiple access levels (also known as tiering). In addition to the access levels, other control information shall be contained which are defined as switches (or flags) that shall indicate specific properties of the PC, the operating system, applications running on the PC, and specific features within the applications.

As an example, the system administrator will have the highest level of access to the operating system including the ability to add or remove applications on the PC, but may not have permissions to run specific applications, functions within the applications, or access data that the applications use.

A PC user may have a lower level of access and cannot alter the operating system and add or remove applications, but the PC user can access the application and the data related to that application.

In another example, corporate rules may apply where only one PC user can be logged on to the PC at any given time and access information on that PC. This requirement is to maintain accountability for the actions taken on the PC such as internet access or manipulation of data that can severely cripple the corporation. In this example, the Proxense utility and driver, and RDC shall only allow only one simultaneous POD to be associated at any given time.

Although the system will enforce access leveling on PC specific contents and operations, it does not imply that when a user with a higher access level enters the cell, their POD disables access to the lower level access user, giving the higher level access user immediate access of the system. Any override operation must be accompanied by a manual interaction of the PC user, either through a button press on the POD, or through the PC utilizing the Proxense utility.

In the scenario where multiple PODs can simultaneously access an RDC the RDC shall provide superframe coordination information to the PODs to interleave them in a manner to avoid contention between the PODs.

As shown in Figure 30, the RDC can assign a superframe count to each POD accessing the RDC. Through link setup and data exchange, the RDC can direct the POD to use a specific superframe or superframes for periodic data exchange. By employing this method, the RDC can alter the wakeup superframe for each POD and can efficiently distribute them to reduce contention between the devices. Since the PODs are given a specific superframe or superframes, the POD is only required to wake up during that superframe(s) to communicate with the RDC.

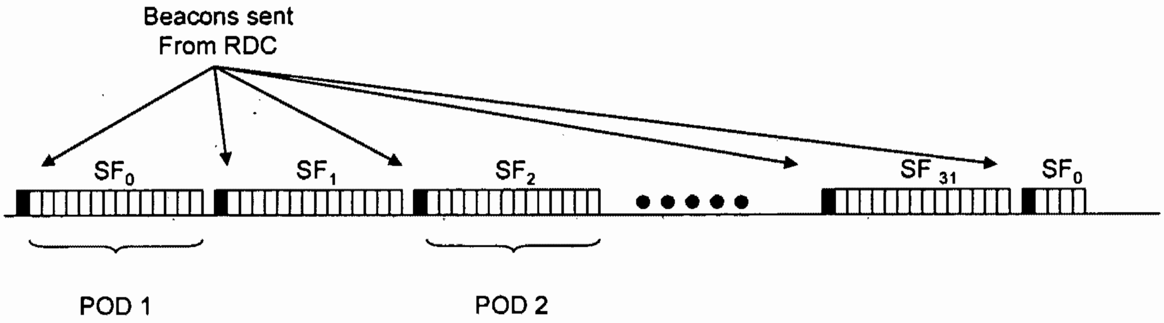


Figure 30: Single cell with multiple POD access

5.3.3 Multi-Cell uncoordinated Operation (Multiple PC Application)

The following description is based on Figure 31 where multiple PCs operating as single cell RDCs are collocated in close proximity with cell overlap. In this configuration, each RDC is not aware of the presence of the other RDC but a POD that resides in the overlap area will detect the presence of both RDCs..

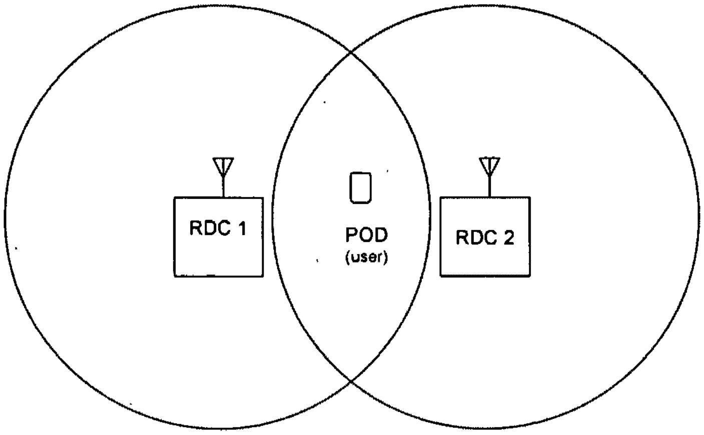


Figure 31: Two Single Cell RDCs with cell overlap

The method of restricting or allowing POD access to one or more RDCs and the method of enabling PC software and hardware components will be governed by the Proxense utility and driver for that specific configuration.

5.3.3.1 Access control

In a configuration where multiple RDCs are collocated without the knowledge of each other, and where a POD is registered to both PCs, a method of determining and gaining access to each PC must be enforced.

There are 2 basic scenarios that can occur. In the first scenario, a PC user may need simultaneous access to both PCs which may be located on the users' desk and where the user is interacting with each PC on a periodic basis. In the second scenario, the PC user may be registered to both PCs, but only wants to associate with only one PC at any given time.

In addition to an RDC granting permission to the POD to access the PC, the RDC may also dictate to the POD the number of associations allowed while associated with that PC. The purpose of defining the number of associations permitted can reduce the possibilities that the POD associates with and enables 2 or more adjacent PCs within close proximity of each other. This prevents a security breach where an individual sitting on another computer, does not gain the ownership of another users PODs access privileges.
The other reason for defining the number of associations is to allow a user to have access to multiple computers in their work environment as previously mentioned.

The capability of configuring the number of associations for a particular RDC will be implemented in the Proxense utility application, driver, and RDC incorporated in the PC.

Referring back to the first scenario, a POD shall be capable of associating to multiple PCs based on the physical limitation of the maximum number of simultaneous connections the POD can handle, and based on the number of associations the RDC permits.

Note: Under the conditions the POD is associated with more than one PC the POD must relay information back to each RDC indicating its timing relative to the other RDC. This information will be very important in the event a single POD is associated with more than one RDC since the clock frequency error between the RDCs will cause eventual timing drift which will eventually causing timeslot and superframe overlap and prevent the POD from communicating with both units on a periodic basis. It should also be taken into consideration where 2 PODs are associated with the same RDC and each POD is also associated with a different second RDC. To further confuse the matter, each of the second RDCs can also have other PODs associated with them that are also associated with even different RDCs. Eventually the uncoordinated system starts looking like a mesh network. A system of this complexity will require a CRDC to be employed to address the synchronization issues.

In the second scenario, there are many methods that may be employed to control how a POD associates with a specific RDC in a PC application. Although there are many methods to employ the system, only 2 are defined here and careful thought and consideration must go into the actual implementation of configuring the RDC relative to the collocation of additional RDCs.

5.3.3.1.1 Method 1 – Association to only one RDC

This is one of the simplest methods where a POD is only associated with a single RDC. Using this method, the POD will attempt to associate to other RDCs, but the other RDCs will deny association causing the POD to ignore the other RDCs as previously explained in section 5.3.2.1.

This method also eliminates the cell size issue, where the cell must be constrained to prevent other RDCs associated with the POD from accessing the POD.

The downside to this method is the fact that the PC user is then locked to a single PC, or PCs that are geographically separated to where the POD cannot see both simultaneously.

5.3.3.1.2 Method 2 – Association by extreme close proximity

This method is the most viable approach to directing a POD to only communicate to one RDC in a configuration where multiple PCs exist of which the POD can associate.

In a configuration where several PCs may be placed in close proximity to each other, by significantly reducing the RF Power level from the RDC and providing this information along with a request for the POD to reduce its RF power, a close proximity

communications channel can be created. The close proximity communications channel can then operate as if only a single cell network exists.
To further explain, if the RDC is configured to have a reduced RF power output, the RDCs cell boundary shrinks causing the POD to be within closer proximity of the RDC to receive a beacon from that RDC. If in turn the RDC indicates in the beacon that it is at reduced RF power the POD is aware that the RDC is in extremely close proximity. In addition, if the beacon contains a command to instruct the POD to reduce RF power, the chance of surrounding RDCs of receiving a response or interference from the POD are minimized.

When the communications channel is terminated and the POD no longer sees the beacon from that RDC, the POD will readjust its RF power level to normal levels for a larger cell coverage area.

The RF power level of the RDC is controllable by the Proxense utility.

5.3.4 Multi-Cell coordinated Operation (Casino Application)

The following description will define the system architecture and the inter-workings of a basic system within a casino application.

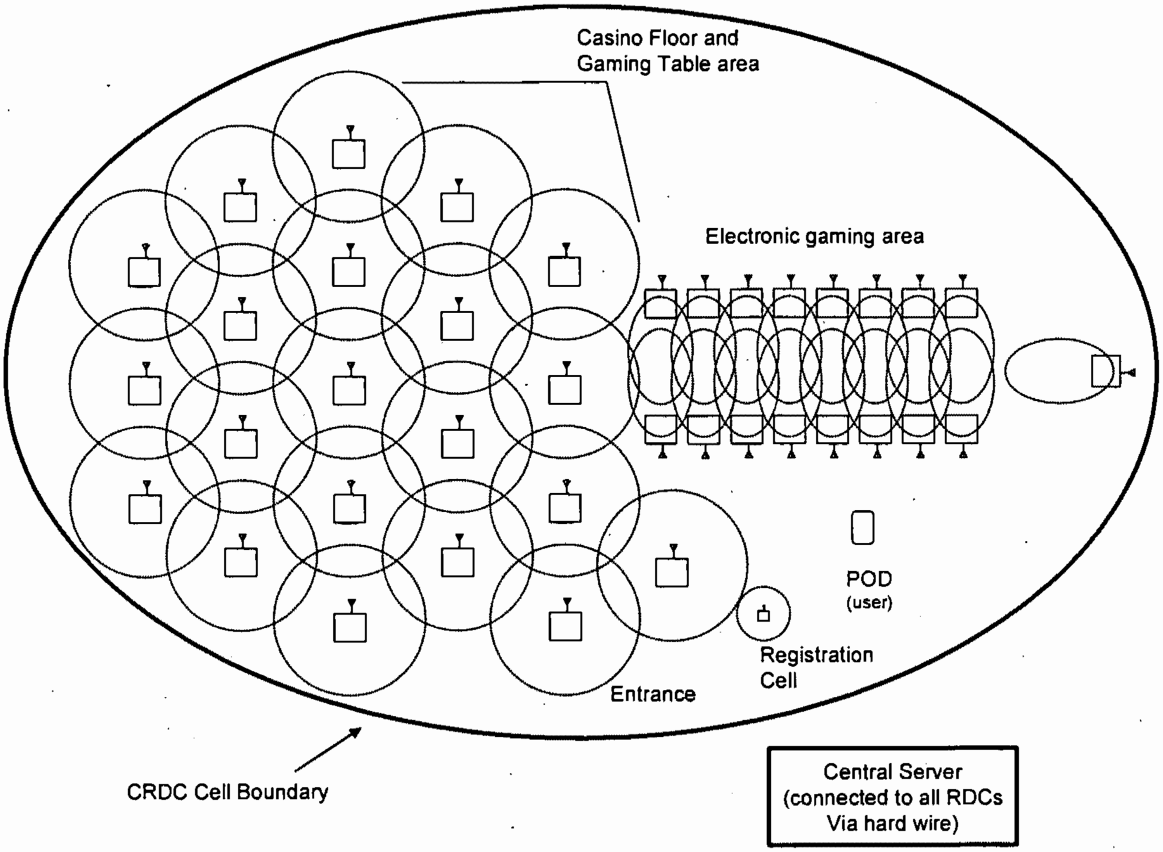

Casino Floor and Gaming Table area
Electronic gaming area
POD (user)
Registration Cell
Entrance
CRDC Cell Boundary
Central Server (connected to all RDCs Via hard wire)

Figure 32: Basic casino floor layout and cell distribution

As shown in Figure 32, a CRDC and multiple RDCs are distributed throughout the casino floor. In this example, only a single CRDC cell exists and give ubiquitous coverage of the entire floor.

On the left side of the figure, multiple RDCs provide overlapped cell coverage and cover the Casino Floor and Gaming table area all the way to the entrance of the casino. This group of RDCs are designated as POD location tracking RDCs and allow the casino operator to know where a player carrying the POD is geographically located on the floor. This group of RDCs may be mounted in the floor or ceiling creating relatively symmetrical cells.

Another set of RDC cells exist in the rightmost part of the diagram and are integrated in the gaming machines within the electronic gaming area. The cell orientation for these RDCs is more oblong and focused at players that are within close proximity and in front of the electronic gaming machines. The cells extend outward towards the center of the isle to detect the presence of a player that may be walking by.

Towards the lower right part of the diagram is a registration cell which sits at a registration desk where a player will register and acquire a POD the first time the player enters the casino.

The registration cell is significantly small in size to allow only local communications between the RDC and POD without allowing external RF monitoring devices to capture and record the interaction between the devices.

A POD is also shown which is currently out of the range of all RDCs, but still in range of the CRDC. The POD is carried by a player to track their position and provide additional services of which some will be discussed below.

Last is a Central Server. The Central Server may contain a player's financial information (credit card numbers, gambling limits, and other information related to a player). In addition, the Central Server if physically wired to all RDCs located throughout the casino.

Although not shown in Figure 32, within the POD location tracking RDCS are gambling tables which also have RDCs imbedded within the table itself. A gambling table may look as shown in Figure 33 below.

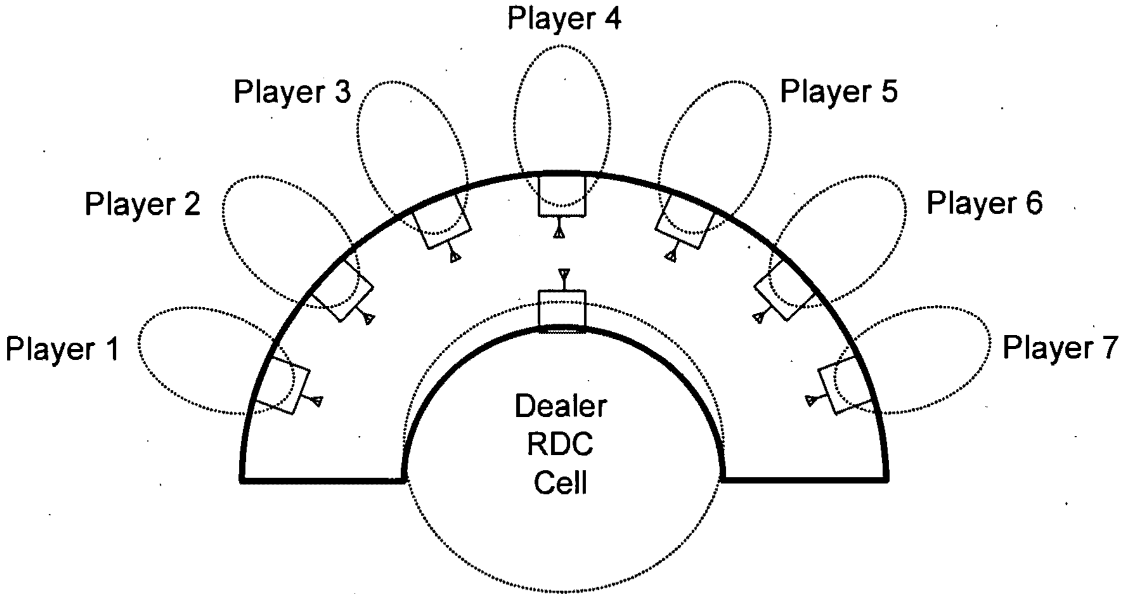


Figure 33: Gambling Table with RDCs example

As shown, the gambling table has 8 RDCs embedded. The first RDC is the dealer RDC which has a cell geometry covering the dealers area allowing the dealer to freely move around in this area.

There are an additional 7 RDCs which are positioned in front of where each player would sit or stand. Each of these RDC cells are oblong and are directional to where the player would be positioned relative to the table.

As shown, the RDC cells are minimized to cover only the area where a dealer or player may be located relative to the table. The actual cells footprint in the final implementation will vary from what is shown above, but will have the same basic affect.

Utilizing distributed RDCs with directional antennas allows the casino operator to know the location of both dealers and players, and the amount of time they remained at the table.

The CRDC, POD location tracking RDCs, and gambling table RDCs, interoperate in the Proxense system.

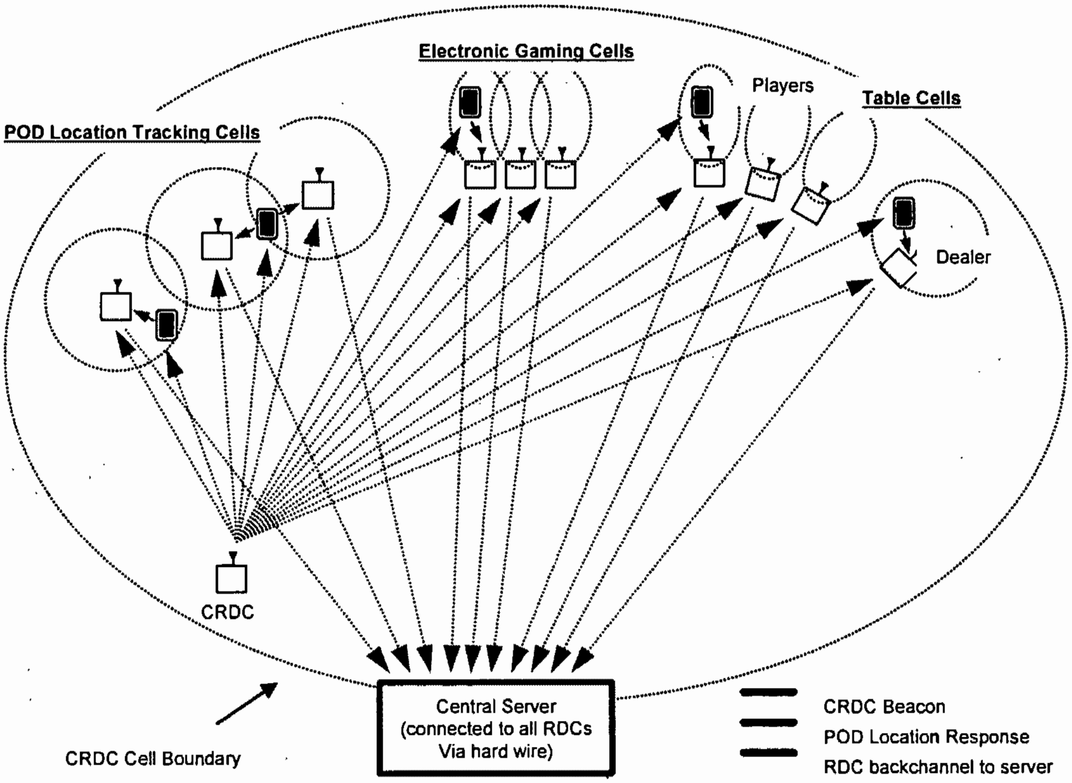

Electronic Gaming Cells
Players
Table Cells
POD Location Tracking Cells
Dealer
CRDC
Central Server (connected to all RDCs Via hard wire)
CRDC Cell Boundary
CRDC Beacon
POD Location Response
RDC backchannel to server

Figure 34: CRDC Beacon to Central Server Flow (POD tracking)

Figure 34 depicts a graphical representation of the sequence of events that occurs when a C-Beacon it transmitted in the casino application. As shown, first the CRDC transmits the C-Beacon to all RDCs and PODs within the CRDCs cell radius. All of the RDCs and PODs setup and synchronize their timing to the beacon. Next, each POD in its appropriate timeslot transmits a POD Location Response ID. Any RDC that is in the local vicinity of the PODs response receives the POD Location Response ID and logs specific information related to the reception. Then each RDC packetizes the information received from the POD and through a wired back channel relays the information to the Central Server.

The Central Server can then take the information, perform a location tracking algorithm, and indicate to the casino operator by either graphical or text format the geographic location of each player.

Representing the flow in a different manner, the basic interactions for a single POD is shown in Figure 35 below.

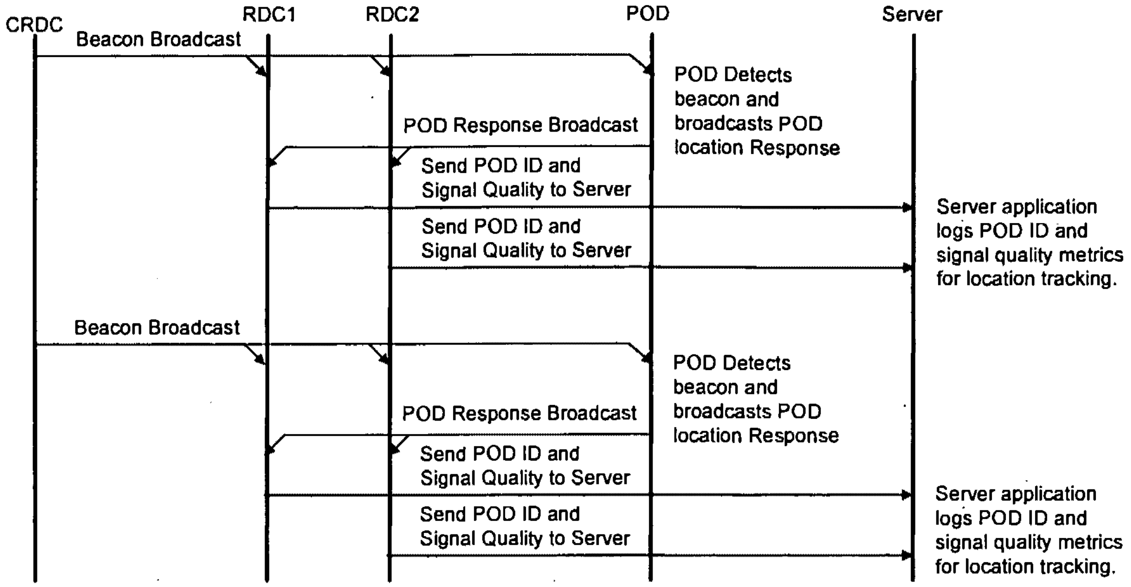

CRDC
RDC1
RDC2
POD
Server
Beacon Broadcast
POD Detects beacon and broadcasts POD location Response
POD Response Broadcast
Send POD ID and Signal Quality to Server
Send POD ID and Signal Quality to Server
Server application logs POD ID and signal quality metrics for location tracking.
Beacon Broadcast
POD Detects beacon and broadcasts POD location Response
POD Response Broadcast
Send POD ID and Signal Quality to Server
Send POD ID and Signal Quality to Server
Server application logs POD ID and signal quality metrics for location tracking.

Figure 35: CRDC Beacon to Central Server Handshake (POD tracking)

In addition to providing location tracking information, when the POD outputs a POD Location Response ID, certain events can occur from an RDC based device. In the event the RDC based device is an electronic game, via the Central Server the electronic game can lure the player over by flashing information on the screen related to that specific player. In one example, an electronic gaming machine can offer the player a free game, luring the player over and ultimately hooking the player on the game. In order to identify the player, further steps are taken between the POD and the RDC which will be described in a later section.

In the event the RDC device is a location RDC, the location RDC can perform further interrogation of the POD to determine the POD is legitimate. A more detailed explanation will follow in another section.

5.3.4.1 Overlapping CRDC Cell configuration

Figure 36 illustrates two CRDCs with overlapping cell coverage, but not to the point where each CRDC can see the other CRDC. As shown the CRDC cells are non-uniform in nature due to obstructions blocking the transmit radio path. The CRDC cell overlap causes a number of RDCs to be in each CRDC cell.

It can be envisioned that other CRDCs can overlap in the same region if the are placed above and below the RDC micro-cellular structure. In this case, up to 4 channels can be occupied just for CRDC beacon transmission.

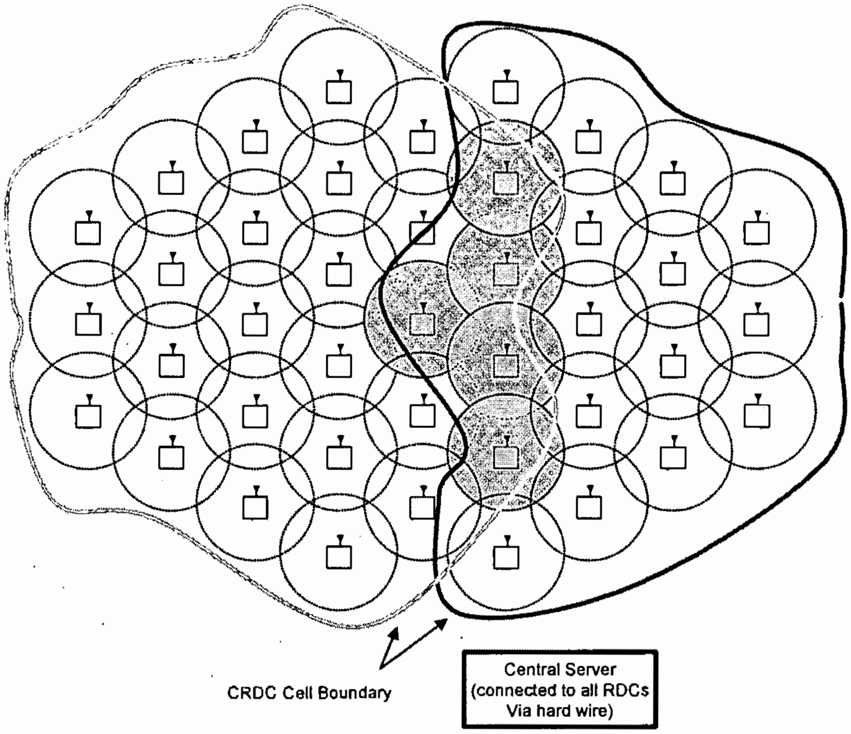


Figure 36: Overlapping CRDC cells

5.3.4.1.1 CRDC Channel selection and assignment

Due to possible interferers in the area of the CRDC cell, the CRDC can be manually configured by the Central Server, or may auto-configure.

5.3.4.1.1.1 Manual Configuration

In the case the Central Server performs manual configuration, upon initial installation and power up, the CRDC is configured to remain in a dormant state until the Central Server interacts with the device and instructs it to perform specific tasks.
The Central Server will first instruct the CRDC to enable the receiver and scan all channels and report back the findings of each channel.

When the CRDC scans each channel it collects signal quality metrics and any 802.15.4 radio transmissions. The operator of the Central Server will analyze all of the signal quality metric information and any 802.15.4 framing information to determine the best channel the CRDC should transmit its beacon on.

One determined, the operator at the Central Server commands the CRDC to store the specific channel that it will transmit on and enable it to begin transmitting.

The operator of the Central Server will then select the next CRDC and perform the same operations, until all CRDCs have been configured in the network.

5.3.4.1.1.2 Automatic Configuration

It is recommended that initial auto-configuration of a CRDC occurs when there is only a single CRDC present. In the auto-configuration mode, the CRDC scans each channel and collects signal quality metrics and determines if other 802.15.4 devices (including another CRDC) is present. The CRDC then selects the quietest channel to begin its beacon transmission on.

The recommendation is based on the fact that the CRDC may have overlapping coverage with another CRDC as illustrated in Figure 36, but the CRDC is not aware of the overlap, causing 2 RDCS to transmit their beacons on the same channel. Eventually due to timing inaccuracies in the CRDCs, they will overlap and become direct interferers to RDCs and PODs in the overlapping area.

The exception to the recommendation, is when the system is configured to allow the CRDC to report back the channel that it occupies (or intends to occupy) to the Central Server, and the Central Server can then analyze the CRDC channel lineup on all CRDCs and reassign a channel for any given CRDC.

Once the channel has been assigned and stored in the local non-volatile memory of the CRDC, upon next power up, the CRDC will scan all channels again and return to their last assigned channel provided it is not occupied by an interferer or another CRDC beacon. In this case, the CRDC need to go through the initialization process again.

5.3.4.1.2 CRDC to CRDC synchronization

Although a well synchronized system leads to higher throughput and can possibly lead to better battery life, based on a preliminary analysis, it was determined the CRDCs are not required to be synchronized. If synchronization is being considered, please refer to Annex B for synchronization options.

Since each CRDC operates on a separate channel, there is no timing considerations that needs to be addressed between CRDCs. The main concern is how the POD aligns to multiple unsynchronized CRDCs.

From a POD location-tracking standpoint, the POD is only required to lock to a single CRDC beacon. The CRDC beacon will indicate to PODs what other channels a CRDC can be located on. The POD does not need to attempt receiving from any other CRDC provided the signal quality metrics for the current CRDC it is monitoring has sufficient signal quality to receive error free data.
In the case the signal quality degrades, the POD will then periodically switch to the other channel(s) to determine if a better signal quality can be obtained. If a better signal quality is determined to be on an alternate channel, the POD will immediately switch to the alternate channel, provided it is not in association with an RDC at that instance in time. If the POD has begun association to an RDC, the POD must attempt to finish the association before switching to the alternate channel.

Note: Once a POD is in association with a RDC, the POD is no longer required to monitor the CRDC beacon, until the association ends between the devices by releasing the link or by the POD leaving the RDC cell.

5.3.4.1.3 RDC Configuration and CRDC selection

When an RDC is powered up for the first time, the RDC is not aware of the network it belongs to or necessarily its configuration within the network. By default, the RDC may resemble a single cell RDC.

For this reason, RDCs placed in a CRDC cell configuration should by default remain in a dormant state upon its first initial power up in the network. This will allow the operator of the Central Server to configure any specific information related to the network into the RDC prior to operation.

When the RDC is powered up the first time, it remains in the dormant state. The operator at the Central Server detects the presence of the RDC and will program specific information into the RDC. Some of this information may include: the site ID, a local RDC ID, and other parameters related to the protocol.

The site ID is important, since the operator does not want the RDC to become associated with another operators site and the RDC should only synchronize to that sites ID.
The RDC ID is used between the communications of a POD and the RDC, and there should only be one RDC in the network containing that RDC ID.
Other parameters may include how the unit operates when associated with a POD and whether the RDC sends data back to the Central Server when data is ready, or if the Central Server must poll the RDC for the information.

The Central Server then commands the RDC to enable its receiver and scan all channels to determine the signal quality of each channel and which CRDCs can be received by the RDC. The operator of the Central Server can allow the RDC to select which CRDC Beacon to lock to, or can command the RDC to automatically select the best channel to receive a CRDC on.

After reviewing the channel list for signal quality, the operator can then command which channels the RDC can use for alternate channels for RDC to POD communications, or the operator can command the RDC to automatically select the alternate channels.

The operator will then place the RDC in operation mode, and the RDC will tune to the selected CRDC Beacon channel and will remain there listening for CRDC Beacons and any POD that sends a POD location Response ID. While receiving the C-Beacon, the RDC will configure its timeslot information similar to that described in Table 10. This will define the superframe structure as well as which timeslots (odd or even) the RDC is permitted to communicate with a POD on, on the beacon channel.

In the background on the alternate Phy, the RDC will continue to scan the alternate channels updating its list of clear channels. This updated list is used when a RDC determines it wants to extend its communications with a POD in association mode, and determine which channel that communications will occur on.

The RDC is now in operational mode performing frame and slot alignment to the CRDC and listening for a POD location tracking response.

On a periodic basis, the RDC will send information back to the Central Server indicating that it is still operational and to the status of the communications channels (CRDC Beacon and alternate channel).

5.3.4.2 Registration RDC configuration

The registration RDC is only used to initially enable and configure a POD into the casinos system. The registration RDC has an extremely small cell coverage area, most likely with a radius in the inches. A POD must be in extreme close proximity if not placed on the registration RDC housing to communicate with the registration RDC.

The registration RDC is connected to the Central Server and other than the specific Proxense security feature, the Central server will install and configure its service provider information of which some can be found in Figure 6 and Figure 7 in section 4.1. The service provider information will include a service provider ID and secret key as well as other parameters that the service provider wants to use for access within their Proxense network. The other parameters can vary in size in the POD and will be defined by the casino operator themselves.

Some of the information transferred to the POD will be:

- The Service Provider Site ID
- The Service Providers assigned POD ID.
- The Service Providers secret service ID.
- The Service Providers secret key.
- And Service Provider specific access information.

5.3.4.3 POD registration and operation

Now that the system has been installed and properly configured, a rated player walks into the casino. They are greeted by a host and walked over to the registration desk, where the player's information, already in the stacking system, is linked to a Proxense PDK® which is given to the player and is assigned with specific privileges.

The player places the POD in their pocket and begins to walk throughout the casino. Once the POD leaves the registration cell, the POD enters discovery mode and scans the channels for a C-Beacon. If the POD doesn't locate the C-Beacon, it continues to scan for an undetermined period of time until it either goes into battery save mode, or finds a C-Beacon.

Once the POD finds the C-Beacon, the POD determines if the C-Beacon is a Proxense network and the Site ID is in its local Service Providers database. If the Site ID is not in the database, the POD ignores that C-Beacon, and keeps looking for other C-Beacons on other channels until one is found that is in its local database.

Once a valid C-Beacon containing a Site ID that is in the PODs local database is found, the POD first extracts the CRDC channel availability flags and checks the other channels in the CRDC channel availability list. The POD then determines which CRDC has the best signal quality metrics.

The POD switches to that channel and begins receiving C-Beacons. The POD extracts the CRDC and network configuration information as described in Sections 5.2.1.2.1 and 5.3.1.1. This information defines the framing structure and how the POD will operate within the network.

The POD then applies the C-Beacon parameters to its radio transceiver parameters configuring the sleep interval and response superframe and timeslot information. Since the POD has just received a C-Beacon, the POD is now aware of the current Superframe count. The POD then configures its timer to wake up just prior to the expected superframe count that it may communicate on.

Referring to Figure 37, when the sleep timer expires the POD will wake up and monitor for its specified C-Beacon and verifies the superframe count. It then waits a predetermined period of time for its slot to be available, and performs CSMA-CA and if no other device is attempting to respond, it responds with its POD location tracking response.

If another device was detected on the channel, the POD will then reset its timers, and wait for the next predefined superframe and timeslot to wake up and attempt again.

When the POD responds with the POD location tracking response, every RDC within local proximity that can receive the response will log the response message in its database along with signal quality metrics and timestamp and will send the information back to the Central Server. The response may be sent by the Central Server polling the RDC, or by the RDC if it has data to send. This must be configured during initial RDC setup.

Once the Central Server receives the POD information from one or multiple RDCs, the Central Server will determine if any further communications with the POD is necessary. If for example, the operator at the Central Server wants to validate the POD, the operator can control the Central Server software to perform a validation. The Central Server will then send a command to a specific RDC to setup communications with a specific POD and wait for an RDC response.

Since the communications between the RDC and Central Server and the action to be taken is not instantaneous, the RDC must wait for the next POD location tracking response and then after performing CSMA-CA immediately instruct the POD to switch to the alternate channel. If during the CSMA-CA the RDC detected another device on the channel, the RDC must wait for the next POD location tracking response from that POD and attempt again.

The POD will switch to the alternate channel and perform CSMA-CA and send a link request to that specific RDC ID with its own ID included.

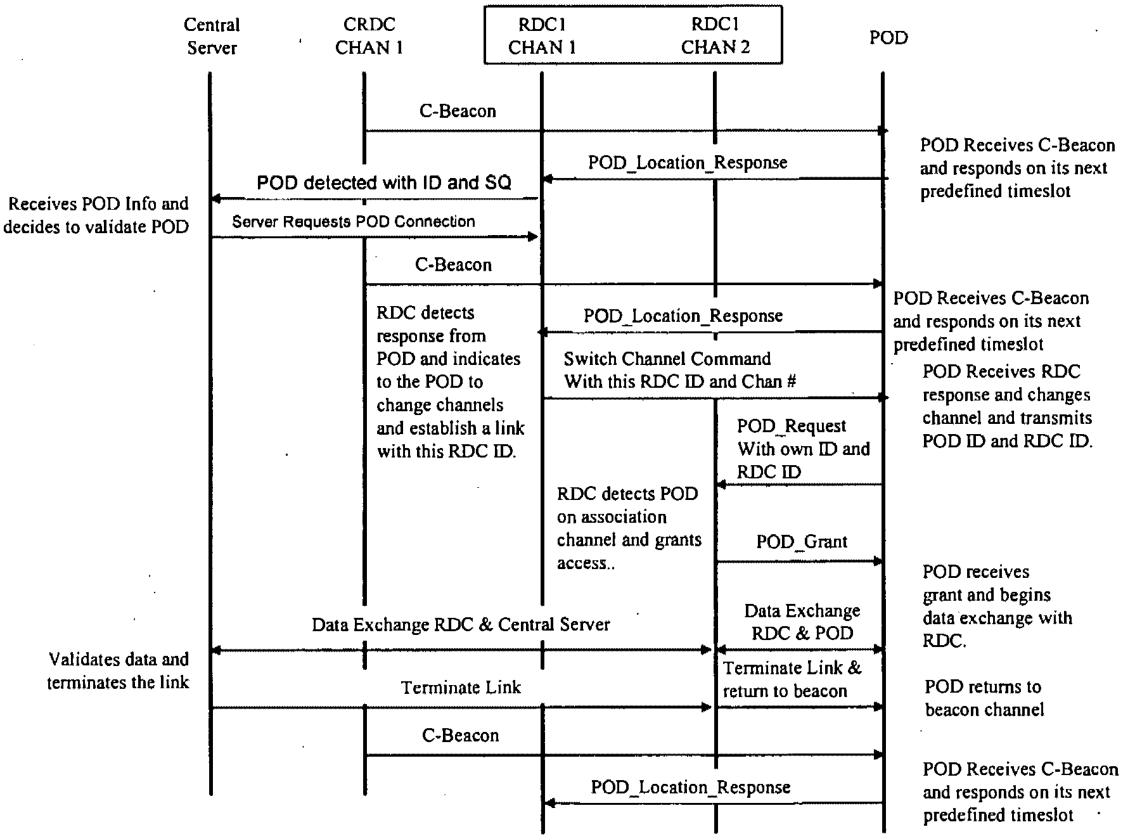

Central Server
CRDC CHAN 1
RDC1 CHAN 1
RDC1 CHAN 2
POD
C-Beacon
POD_Location_Response
POD Receives C-Beacon and responds on its next predefined timeslot
POD detected with ID and SQ
Receives POD Info and decides to validate POD
Server Requests POD Connection
C-Beacon
POD Receives C-Beacon and responds on its next predefined timeslot
RDC detects response from POD and indicates to the POD to change channels and establish a link with this RDC ID.
POD_Location_Response
Switch Channel Command With this RDC ID and Chan #
POD Receives RDC response and changes channel and transmits POD ID and RDC ID.
POD_Request With own ID and RDC ID
RDC detects POD on association channel and grants access..
POD_Grant
POD receives grant and begins data exchange with RDC.
Data Exchange RDC & Central Server
Data Exchange RDC & POD
Validates data and terminates the link
Terminate Link
Terminate Link & return to beacon
POD returns to beacon channel
C-Beacon
POD_Location_Response
POD Receives C-Beacon and responds on its next predefined timeslot

Figure 37: C-Beacon handoff to RDC to POD communications

The RDC looking for the link request with a specific RDC ID and the POD ID detects the request and then responds immediately with a link grant.

The RDC alerts the Central Server of the link and a data exchange occurs of the information the Central Server is interested in interrogating. After the exchange occurs, the Central Server commands the RDC to terminate the link. The RDC then terminates the link and the POD returns back to the C-Beacon channel re-synchronizing to the channel and monitoring for its timeslot. The POD then continues to send responses back to all RDCs when its specific superframe count and timeslot are valid.

Note: Although this example indicated a validation of the POD, the Central Server could have altered the Service Provider information within the POD.

If during the switch to the alternate channel for RDC to POD communications, the POD determines the channel is occupied, or the POD does not receive a link grant back from the RDC, the POD will perform 2 more attempts. If after all 3 attempts the POD does not receive a response, the POD will return to the C-Beacon channel, realign to the beacon, and then begin sending its POD location tracking ID.

If the RDC was unable to receive the link request from the POD on the alternate channel, for a predefined period of time, the RDC will flag the error and continue listening on the channel.

If the same RDC is again instructed to establish communications with the same POD, the RDC may choose to use a different alternate channel and redirect the POD to the new alternate channel for communications.

Figure 38 below illustrates the POD wakeup and response flow in a CRDC coordinated system. It is assumed that the POD has acquired system synchronization and has gone into the sleep mode after settings its timers to wake up on the next predefined superframe. The POD remains in sleep mode until the wakeup timer expires and wakes up the POD. The POD then enables and tunes its receiver to the C-Beacon channel and listens for the beacon for a predefined period of time. If no beacon is detected, the POD looks for the other CRDC channel available flags and reassesses signal quality and Beacons on the alternate CRDC channels.

After the POD access the other CRDC channels if no channel is found, the POD goes back into rediscovery mode scanning all channels looking for a C-Beacon. If no C-Beacon is found, the POD then starts its deep sleep mode.

In the event other CRDC channels are present, the POD assesses the signal quality of each channel and selects the best channel. The POD then selects that channel and tunes to it listens for the CRDC beacon.

When the POD receives the beacon, it checks all of the parameters associated with it including the superframe count. If the POD determines the superframe count it not the correct one for it to wake up and respond on, it sets its internal sleep timer to wake up just before the next expected superframe it should respond to and returns to sleep mode.

If the POD determines that the beacon is on the PODs expected superframe count, the POD then stays awake but stops listening until just before its expected timeslot. The POD then performs CSMA-CA to determine if the channel is busy. If the POD determines the channel is busy, the POD will again set its internal sleep timer for the next expected superframe and return to the sleep mode.

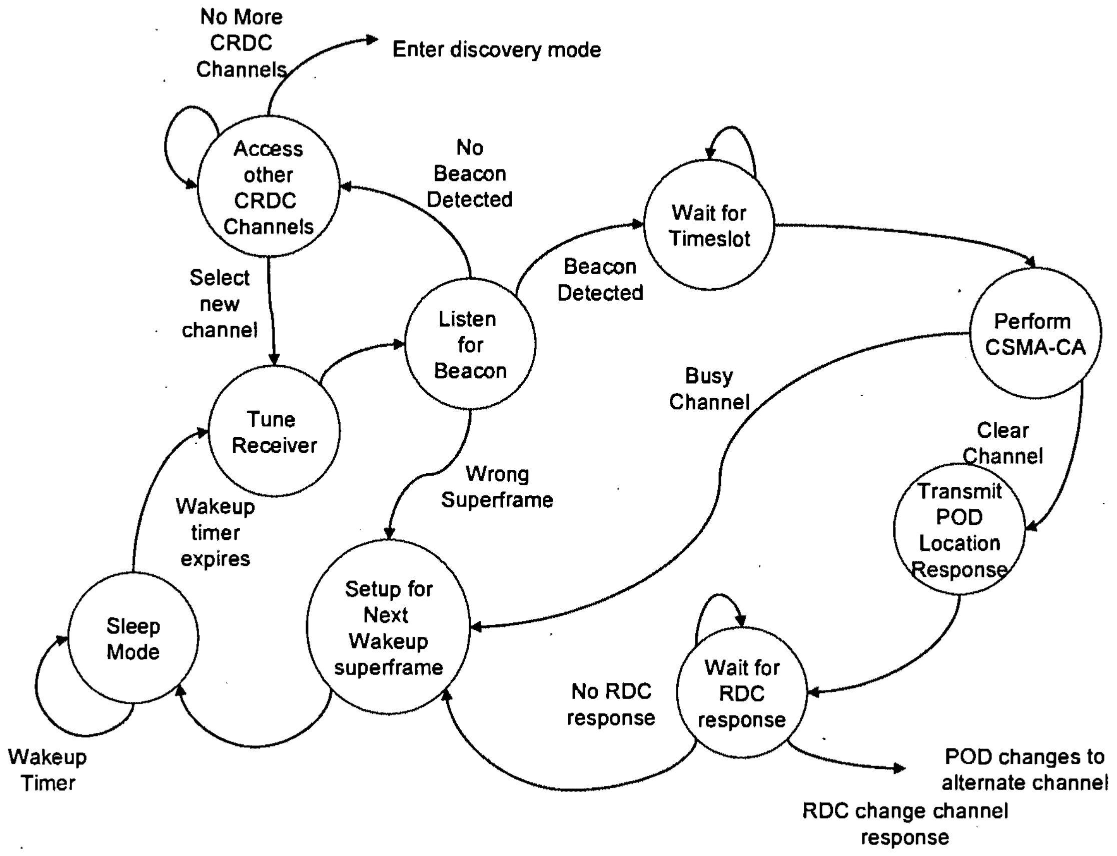


Figure 38: POD wakeup and response state flow

If the POD finds the channel to be available, the POD then transmits its POD location tracking response and waits for one additional timeslot for a response from an RDC.

If the POD receives a response from an RDC, it then performs the command sent by the RDC (in most cases to switch to the alternate communications channel).

If the POD does not receive a response from an RDC, the pod again sets its internal sleep timer to wake up just before the next expected superframe it should respond to and returns to sleep mode.

5.3.4.4 Electronic Game with integrated RDC configuration

As briefly mentioned in section 5.3.4, the RDC is located within the electronic games on the electronic gaming floor.

Each electronic game will have integrated a dual phy RDC. The RDC is used for POD location tracking and POD association.
There are two basic approaches in integrating an RDC into an electronic game. The basic approaches are described below.

5.3.4.4.1 Integrated RDC with no internal game interconnect

Figure 39 illustrates an electronic game with an integrated RDC. The integration of the RDC is from a physical perspective only and no electric connections exist between the RDC and electronic game.

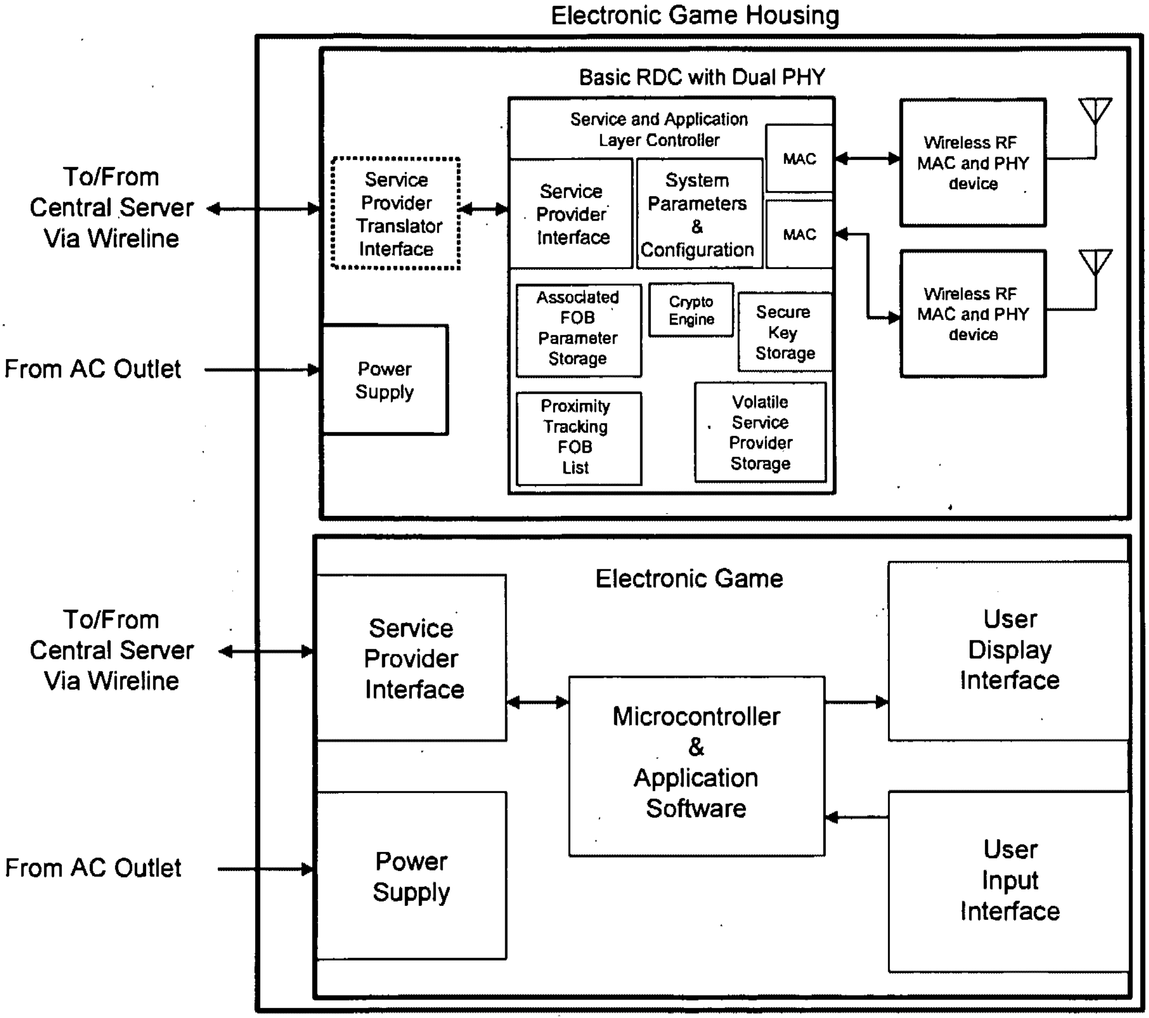

Electronic Game Housing
Basic RDC with Dual PHY
To/From Central Server Via Wireline
Service Provider Translator Interface
Service and Application Layer Controller
Service Provider Interface
System Parameters & Configuration
MAC
MAC
Wireless RF MAC and PHY device
Wireless RF MAC and PHY device
Associated FOB Parameter Storage
Crypto Engine
Secure Key Storage
Proximity Tracking FOB List
Volatile Service Provider Storage
From AC Outlet
Power Supply
Electronic Game
To/From Central Server Via Wireline
Service Provider Interface
Microcontroller & Application Software
User Display Interface
User Input Interface
From AC Outlet
Power Supply

Figure 39: Electronic Game with integrated RDC (no internal connections)

In this configuration, both the RDC and electronic game reside within the same game enclosure and they coexist as two devices. They are not connected in any way other than physical mounting.

The purpose of placing them in the same enclosure is to allow the RDC to perform proximity detection for any player carrying a POD that may be positioned near the front of the machine.

In this configuration each device (RDC and electronic game) have separate connections to the Central Server or an external combining device can be used to connect both devices to a single wired connection back to the Central Server.

The electronic game will operate normally with the exception of any commands sent by the Central controller. The RDC will provide both proximity detection and association with PODs.

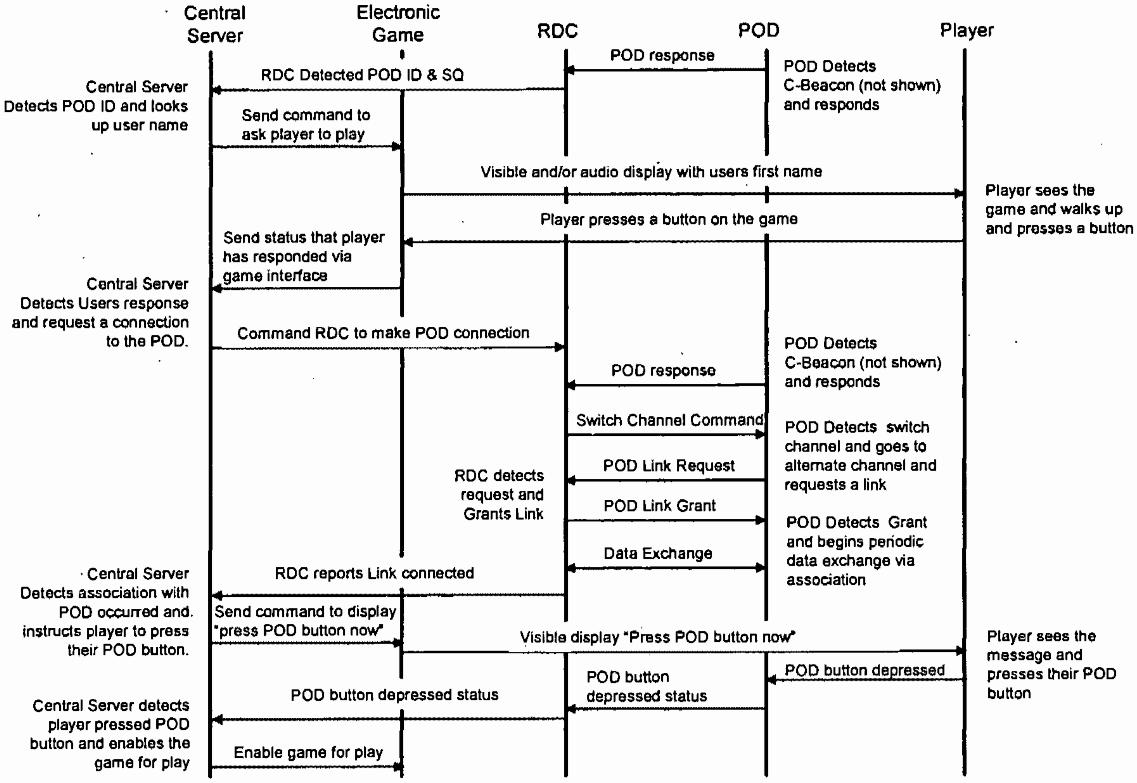

Central Server
Electronic Game
RDC
POD
Player
POD response
POD Detects C-Beacon (not shown) and responds
RDC Detected POD ID & SQ
Central Server Detects POD ID and looks up user name
Send command to ask player to play
Visible and/or audio display with users first name
Player sees the game and walks up and presses a button
Player presses a button on the game
Send status that player has responded via game interface
Central Server Detects Users response and request a connection to the POD.
Command RDC to make POD connection
POD Detects C-Beacon (not shown) and responds
POD response
Switch Channel Command
POD Detects switch channel and goes to alternate channel and requests a link
POD Link Request
RDC detects request and Grants Link
POD Link Grant
POD Detects Grant and begins periodic data exchange via association
Data Exchange
Central Server Detects association with POD occurred and. instructs player to press their POD button.
RDC reports Link connected
Send command to display "press POD button now"
Visible display "Press POD button now"
Player sees the message and presses their POD button
POD button depressed
POD button depressed status
POD button depressed status
Central Server detects player pressed POD button and enables the game for play
Enable game for play

Figure 40: Electronic Game Player tracking and game enable handshake

To reduce complexity of the diagram, the CRDC is not shown.

As shown in Figure 40, the basic handshake that could take place from the time a player carrying a POD is detected, to the time the game is enabled for that player is shown.

The handshake starts by the POD detecting a C-Beacon. Each time the C-Beacon is detected on the expected superframe and timeslot, the POD sends out a POD location tracking response.

The RDC within the game detects the response and sends the POD information back to the Central Server. The Central Server realizes the user is close to the game, and sends a command back to the game instructing it to display a message for the player "Welcome Joe, if you'd like to play a free game press the start button".

Joe sees the message and sits down at the game and presses the button. In turn, the game sends a message back to the Central Server indicating that the button has been pressed.

The Central Server then requests the RDC to make a connection with Joes POD. Upon the next C-Beacon, Joes POD responds and the RDC receives the response. The RDC then immediately transmits back to Joes POD to change to another channel for association.

Joes POD switches to the alternate channel and sends out a POD link request with the POD ID and the RDC ID. The RDC detects the request and sends back a POD link grant.

The POD and the RDC then exchange some security information to establish a secure link for validation of the POD. The RDC may also lower its RF power and request the POD to lower its RF power to guarantee close proximity. Periodically data exchange continues between the RDC and POD.

After the secure link was established, the RDC reports back to the Central Server that the link is established between the RDC and POD. The Central Server then sends a command to the game to display "Press your POD button for your free game" of which the game displays.

The user sees the message on the screen and presses their button causing the POD to transmit the button press over the secure link to the RDC. The RDC in turn sends the information back to the Central Server.

When the Central Server receives the message that the POD button was depressed, the Central Server enables the game and the player begins playing.

The handshake continues in Figure 41 below. After the game was enabled for Joe to play, the Central server sends a command to the RDC to start polling the POD.
The RDC then periodically polls the POD and optionally returns the response of each poll to the Central Server (shown in green). Whether the response is optional or mandatory is based on the operators' preferences and the command sent to the RDC.

Joe continues to play the game for a while and then finishes and decides to leave. When Joe leaves the near proximity of the game, the communications link is broken. The RDC attempts to poll the POD but receives no response. The RDC continues 2 more times with no response. The RDC then reports back to the Central Server that the link was lost and the POD is out of range.

The Central Server immediately returns the game to an idle state so another player can play and requests the game to send it the statistics for Joe. The Central Controller then logs the statistics.

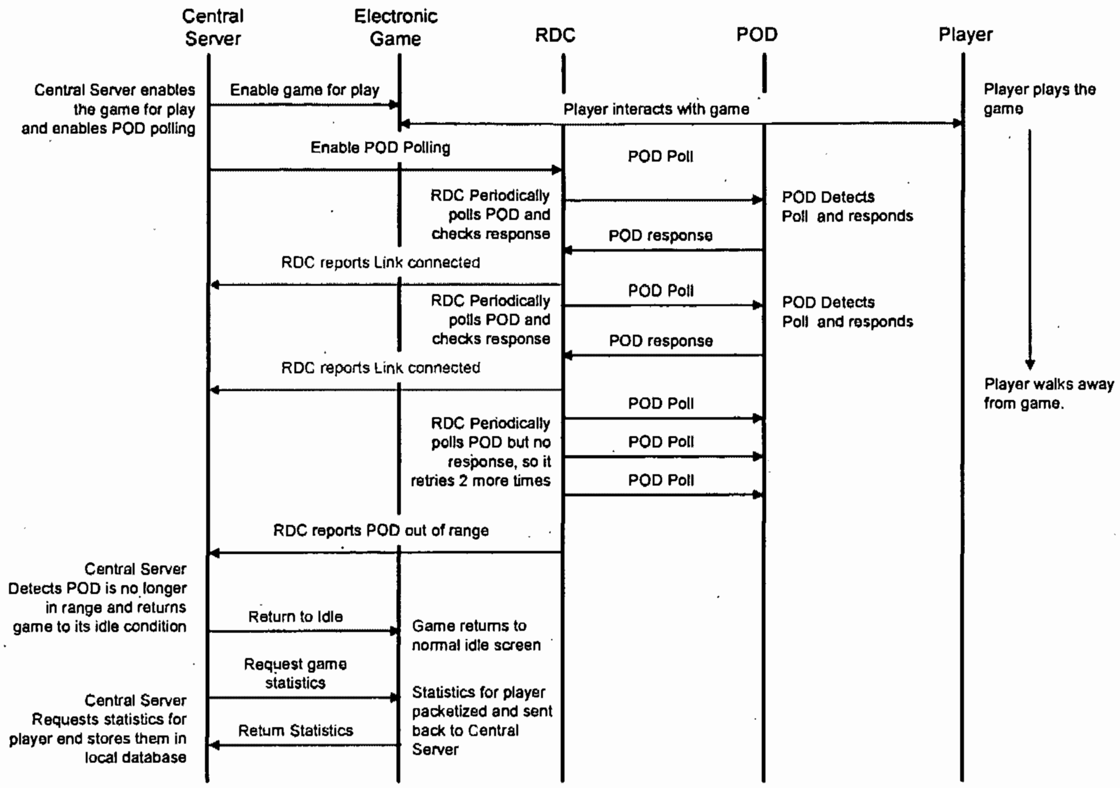

Central Server
Electronic Game
RDC
POD
Player
Central Server enables the game for play and enables POD polling
Enable game for play
Player interacts with game
Player plays the game
Enable POD Polling
POD Poll
RDC Periodically polls POD and checks response
POD Detects Poll and responds
POD response
RDC reports Link connected
POD Poll
RDC Periodically polls POD and checks response
POD Detects Poll and responds
POD response
RDC reports Link connected
Player walks away from game.
POD Poll
RDC Periodically polls POD but no response, so it retries 2 more times
POD Poll
POD Poll
RDC reports POD out of range
Central Server Detects POD is no longer in range and returns game to its idle condition
Return to Idle
Game returns to normal idle screen
Request game statistics
Central Server Requests statistics for player end stores them in local database
Statistics for player packetized and sent back to Central Server
Return Statistics

Figure 41: Electronic Game Player association handshake

As described above the Central Server is the median between the RDC and the game to enable the game and tie the POD to that game. If the connection was lost to the central server by either device, there is a possibility that the game may stay enabled.

5.3.4.4.2 Integrated RDC with internal game interconnect

Figure 42 illustrates an electronic game with an integrated RDC that the 2 devices are electrically connected. All power and communications for the RDC goes through the electronic game.

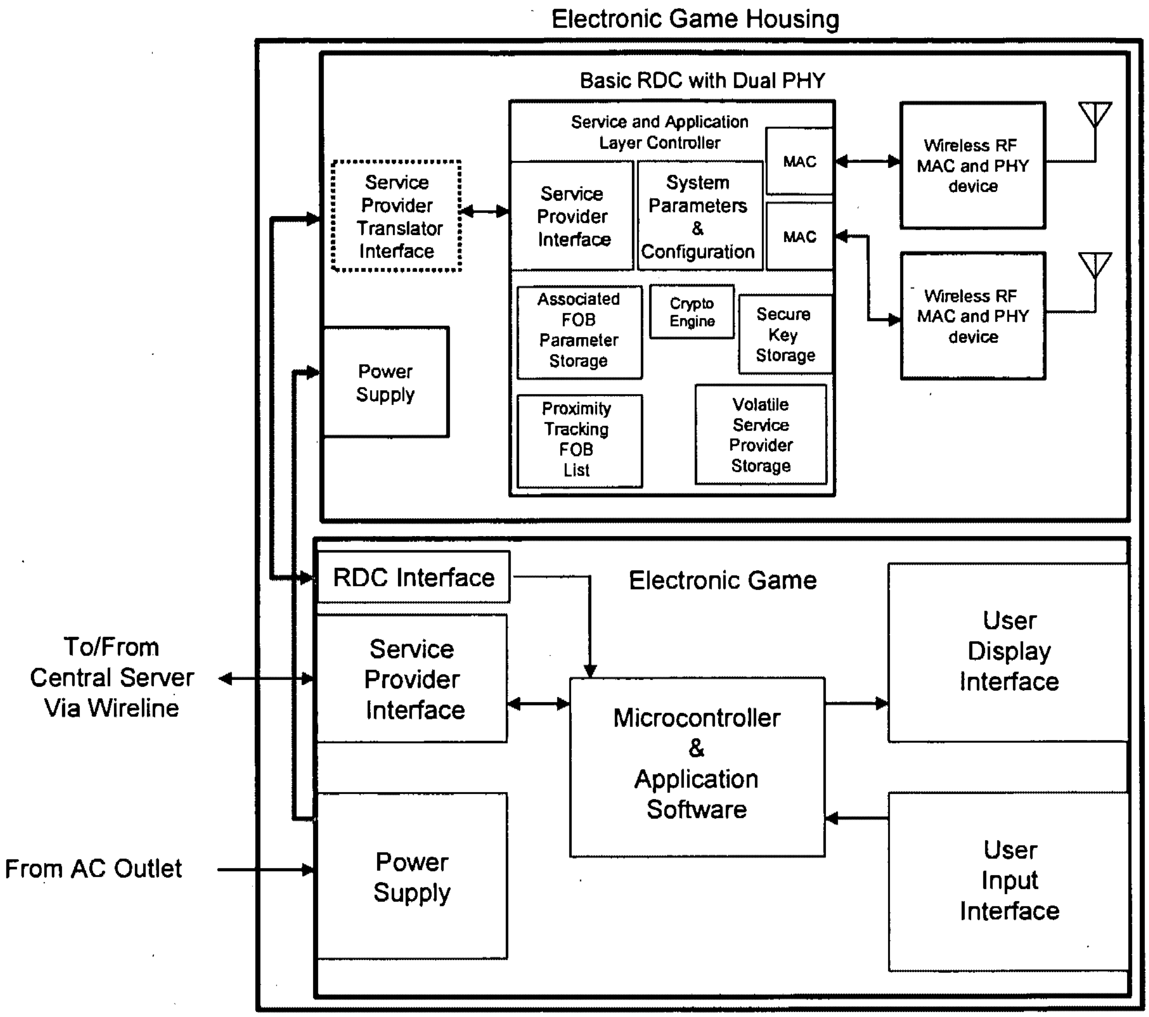

Electronic Game Housing
Basic RDC with Dual PHY
Service and Application Layer Controller
MAC
MAC
Wireless RF MAC and PHY device
Wireless RF MAC and PHY device
Service Provider Translator Interface
Service Provider Interface
System Parameters & Configuration
Associated FOB Parameter Storage
Crypto Engine
Secure Key Storage
Power Supply
Proximity Tracking FOB List
Volatile Service Provider Storage
RDC Interface
Electronic Game
User Display Interface
To/From Central Server Via Wireline
Service Provider Interface
Microcontroller & Application Software
From AC Outlet
Power Supply
User Input Interface

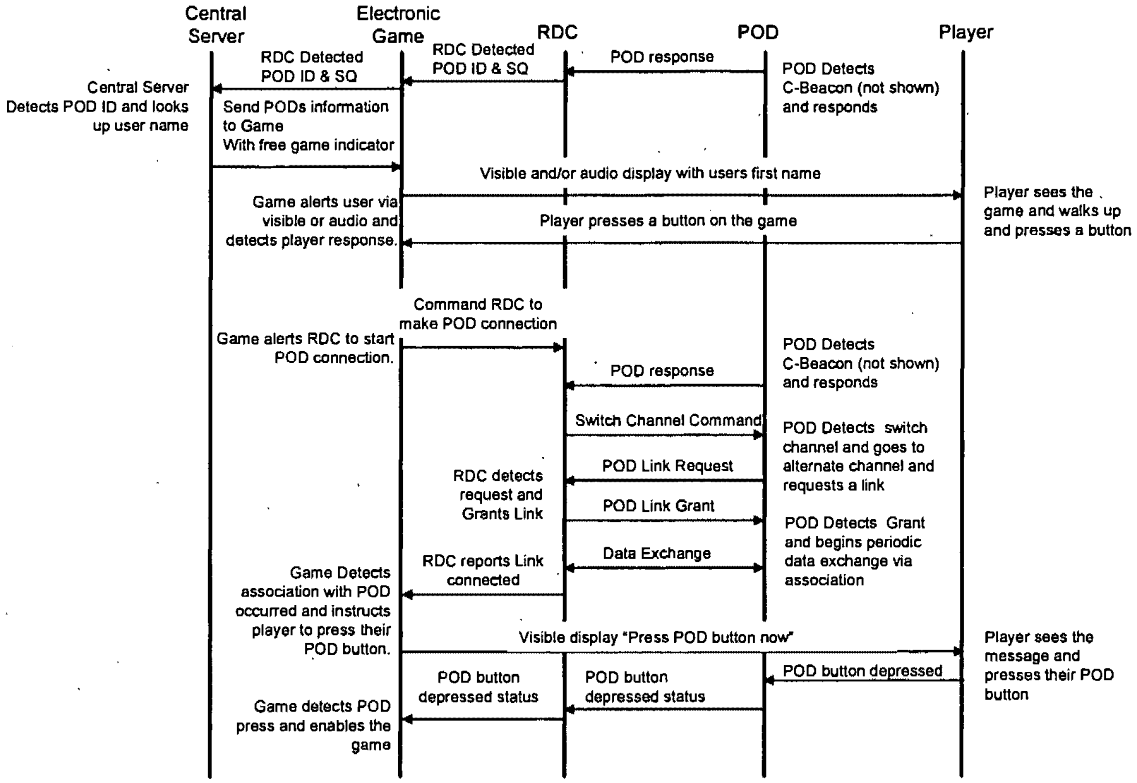
Figure 42: Electronic Game with integrated RDC (internally connected)

In this configuration, both the RDC and electronic game reside within the same game enclosure and they jointly coexist. If information must flow to and from the Central Server from the RDC, the communications must pass through the electronic game controller.

The purpose of placing them in the same enclosure is the same as in the previous description. This allows the RDC to perform proximity detection for any player carrying a POD that may be positioned near the front of the machine.

The primary difference in this configuration versus the previous configuration is the electronic game can offload the Central Server and perform more local verification of the communications link between the POD and the RDC.

To illustrate the difference in interaction between for this configuration, refer to the handshake diagram shown in Figure 43.



Central Server
Electronic Game
RDC
POD
Player
RDC Detected POD ID & SQ
RDC Detected POD ID & SQ
POD response
POD Detects C-Beacon (not shown) and responds
Central Server Detects POD ID and looks up user name
Send PODs information to Game With free game indicator
Visible and/or audio display with users first name
Player sees the game and walks up and presses a button
Game alerts user via visible or audio and detects player response.
Player presses a button on the game
Command RDC to make POD connection
Game alerts RDC to start POD connection.
POD response
POD Detects C-Beacon (not shown) and responds
Switch Channel Command
POD Detects switch channel and goes to alternate channel and requests a link
POD Link Request
RDC detects request and Grants Link
POD Link Grant
POD Detects Grant and begins periodic data exchange via association
Data Exchange
RDC reports Link connected
Game Detects association with POD occurred and instructs player to press their POD button.
Visible display "Press POD button now"
Player sees the message and presses their POD button
POD button depressed
POD button depressed status
POD button depressed status
Game detects POD press and enables the game

Figure 43: Electronic Game Player tracking and game enable handshake via game

To reduce complexity of the diagram, the CRDC is not shown.

The basic handshake that could take place from the time a player carrying a POD is detected, to the time the game is enabled for that player is shown.

The handshake starts by the POD detecting a C-Beacon. Each time the C-Beacon is detected on the expected superframe and timeslot, the POD sends out a POD location tracking response.

The RDC within the game detects the response and sends the POD information back to the Central Server via the game controller. The Central Server realizes the user is close to the game, and sends a command back to the game controller instructing the game to give the user a free game along with the users' information. The game then displays a message for the player "Welcome Joe, if you'd like to play a free game press the start button".

Joe sees the message and sits down at the game and presses the button. In turn, the game controller detects the button press and requests the RDC to make a connection with Joes POD. Upon the next C-Beacon, Joes POD responds and the RDC receives the response. The RDC then immediately transmits back to Joes POD to change to another channel for association.

Joes POD switches to the alternate channel and sends out a POD link request with the POD ID and the RDC ID. The RDC detects the request and sends back a POD link grant.

The POD and the RDC then exchange some security information to establish a secure link for validation of the POD. The RDC may also lower its RF power and request the POD to lower its RF power to guarantee close proximity. Periodically data exchange continues between the RDC and POD.

After the secure link was established, the RDC reports back to the game controller that the link is established between the RDC and POD. The game then displays "Press your POD button for your free game".

The user sees the message on the screen and presses their button causing the POD to transmit the button press over the secure link to the RDC. The RDC in turn sends the information back to the game controller.

The game controller then enables the game and the player begins playing.

The handshake continues in Figure 44 below. After the game was enabled for Joe to play, the game controller sends a command to the RDC to start polling the POD.
The RDC then periodically polls the POD and optionally returns the response of each poll to the game controller (shown in green). Whether the response is optional or mandatory is based on the operators' preferences and the command sent to the RDC.

Joe continues to play the game for a while and then finishes and decides to leave. When Joe leaves the near proximity of the game, the communications link is broken. The RDC attempts to poll the POD but receives no response. The RDC continues 2 more times with no response. The RDC then reports back to the game controller that the link was lost and the POD is out of range.

The game controller immediately returns the game to an idle state so another player can play and indicates back to the Central Server that the POD is out of range. The Central Server then requests the game to send it the statistics for Joe where it logs Joe's statistics.

As described above the game controller becomes more involved in the RDC and POD association and removes the overhead from the Central Server. The game controller can also react faster to the user walking out of range and does not require any response from the Central Server to maintain the link.

The issue with a broken link between the Central Server, game controller, and RDC are now resolved, but under the conditions of a power failure the operator may lose any information related to the game play if the game or its memory does not have battery backup.

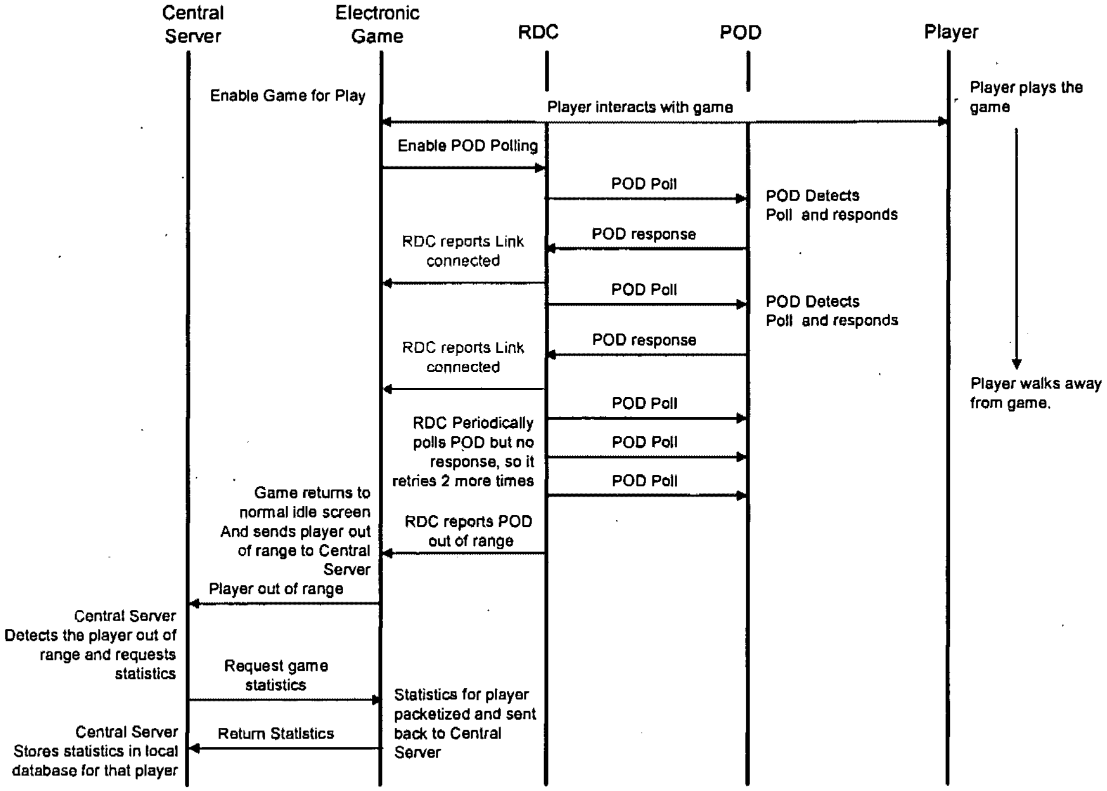

Central Server
Electronic Game
RDC
POD
Player
Enable Game for Play
Player interacts with game
Player plays the game
Enable POD Polling
POD Poll
POD Detects Poll and responds
RDC reports Link connected
POD response
POD Poll
POD Detects Poll and responds
RDC reports Link connected
POD response
Player walks away from game.
POD Poll
RDC Periodically polls POD but no response, so it retries 2 more times
POD Poll
POD Poll
Game returns to normal idle screen And sends player out of range to Central Server
RDC reports POD out of range
Player out of range
Central Server Detects the player out of range and requests statistics
Request game statistics
Statistics for player packetized and sent back to Central Server
Central Server Stores statistics in local database for that player
Return Statistics

Figure 44: Electronic Game Player association handshake via game

It should also be noted that the game controller software will need modification to provide the interaction with the RDC and to allow passing of data between the RDC and the Central Controller.

6 RDC Wired Protocol

Up until this point, the primary focus has been the wireless protocol with brief mentions of the backend wired protocol that connects to a service provider or other equipment such as a personal computer.

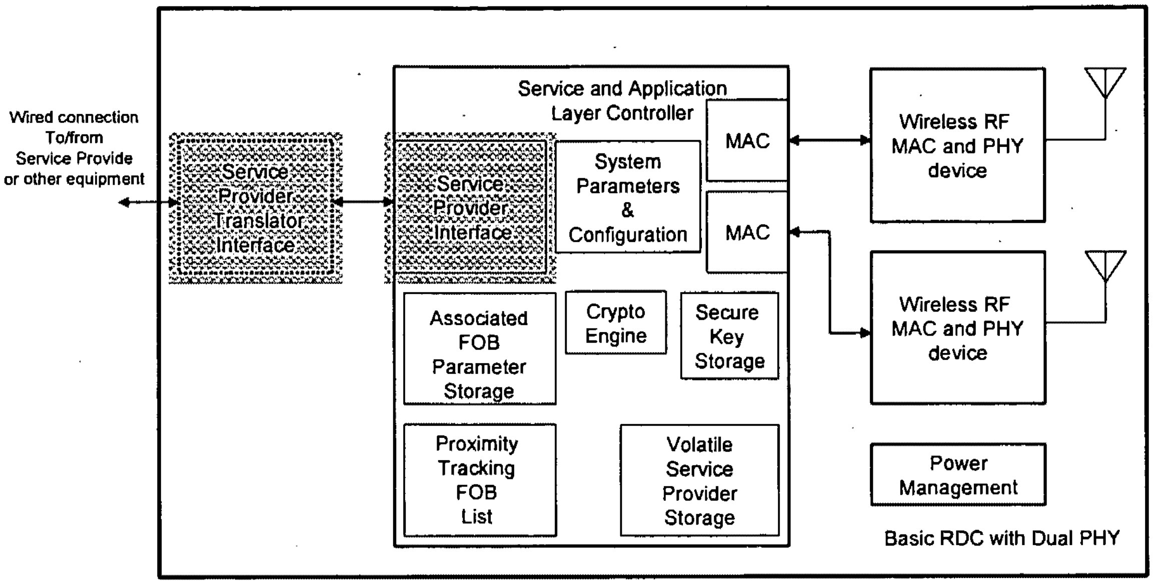

Wired connection
To/from
Service Provide
or other equipment
Service Provider Translator Interface
Service Provider Interface
Service and Application Layer Controller
System Parameters & Configuration
MAC
MAC
Wireless RF MAC and PHY device
Wireless RF MAC and PHY device
Associated FOB Parameter Storage
Crypto Engine
Secure Key Storage
Proximity Tracking FOB List
Volatile Service Provider Storage
Power Management
Basic RDC with Dual PHY

Figure 45: Basic RDC block diagram with wired connection side highlighted

In the diagram shown in Figure 45, 2 specific blocks are highlighted. The light green block is part of the Proxense proprietary design and provides the wired interface to the Proxense chipset.

The light blue block is not part of the Proxense proprietary chipset but is inserted in this diagram to allow for easy system translation of the Proxense proprietary chipset protocol to the native system protocol. This block also allows for various physical interfaces based on other chipset users requirements.

6.1 Proxense chipset interface

The Proxense chipset interface is broken down into three parts. The first part is the physical layer that defines the electro-mechanical interface, the second is the transport layer that defines raw formatting of data transmitted, and the third part is the Service Layer that defines how data is handled.

6.1.1 Proxense chipset physical layer

As of this time, the physical layer for the wired protocol has not been finalized. There are several options that are under consideration, and this section will discuss each of these options.

For reference purposes in this section, the external controller that will connect to the Proxense chipset will be referred to as the master, and the Proxense chipset will be referred to as the slave.

6.1.1.1 Synchronous Serial Peripheral Interface (slave)

The Synchronous Serial Peripheral Interface (SPI), is a 4 wire synchronous interface which allows for high speed full duplex communications with low pin count. Since the interface is synchronous, the clock is sent with the data that does not require a master or slave to synchronize to the data stream.

The SPI interface only defines the physical interface and does not have a transport protocol.

Figure 46 illustrates the SPI interconnect. The primary SPI interface consists of four wires of which four signals are for communications and two signals for control/status.

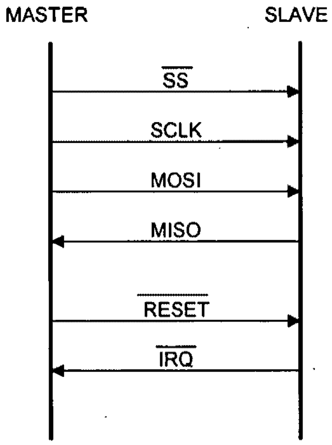


Figure 46: Synchronous Serial Peripheral Interface Connections

The signals shown in the diagram are defined as follows:

- /SS – Slave Select (low true) – this signal is the enable signal for the slave device when driven to a logic low level. When driven to a logic high level, the slave device is not selected. This signal is used to select the device to communicate with and must stay low during the data transfer period.

- SCLK – Serial Clock – This signal is the source clock that the serial data will align to. In most microcontrollers the phase and polarity of the signal is programmable relative to the data output and capture. A normal configuration would be to clock data out on the rising edge, and capture data on the falling edge.
- MOSI – Master Out Slave In – This is the serial data line that transmits data from the master and receives data at the slave.
- MISO – Master In Slave Out – This is the serial data line that transmits data from the slave and received data at the master.
- /RESET – Reset (low True) – This signal is driven by the master and received at the slave. When driven to a logic low level, the slave device is held in a reset state. When driven to a logic high level, the slave device is operational.
- /IRQ – Interrupt Request (low true) – This signal is driven from the slave device and received at the master device. Another name for this signal might be "attention", since the slave device uses this line to gain the attention of the master that it is requesting to be serviced.

Figure 47 shows a basic one-byte transfer using the SPI interface. As shown, the slave select line will be driven low by the master to enable the slave device to receive the clock and any data. On each rising edge of the clock (also driven by the master), the data is driven out of each device onto the serial bus, and on each falling edge of the clock the receiving device captures data. Data is always sent msb first.

When the transfer is complete, the slave select line is returned to a logic high level where the slave device is deselected and will not respond to any transitions on the clock line until the next time the slave select is driven low. Each device will maintain the same value on the serial data output pin after the last rising edge clock until the first clock of the next transmission, or if either device has tri-stating capabilities, it will place the lines in high impedance when the slave select returns to a logic high state.

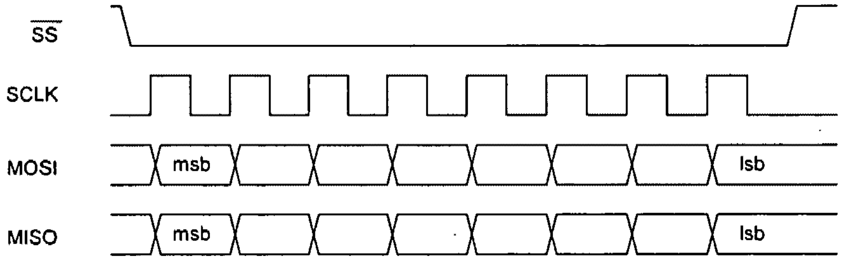


Figure 47: Synchronous Peripheral Interface Waveform diagram

To perform multiple byte transfers, the slave select line must remain low and the master must continue to clock the SCLK line. In most SPI controllers, any time data is present in the transmit FIFO a byte is serial transmitted on the SPI bus with 8 clocks being generated for that byte.

If there is no data to transmit from the master but data is to be received from the slave, the software must still place a dummy byte into the transmit FIFO to generate the clocks to receive a byte from the slave.

There are 3 distinct advantages to this approach. First, there is no tolerance requirement between the master and slaves reference clocks since a clock is provided with the serial bit stream. Second, any data rate can be achieved to the maximum the master can generate and the slave can receive. And third, multiple slave devices can be attached to a master provided the slave devices tristate the MISO output.

One distinct disadvantage is the master must be involved in any transfers from the slave.

6.1.1.2 Inter-IC (I2C) Bus (slave)

The I2C bus is a 2 wire synchronous interface that allows for moderate speed half duplex communications with low pin count. Since the interface is synchronous, the clock is sent with the data that does not require a master or slave to synchronize to the data stream.

The I2C bus defines both a physical interface and transport protocol.

Figure 48 illustrates the I2C interconnect. The primary I2C interface consists of two wires of which two signals are for communications and two signals for control/status.

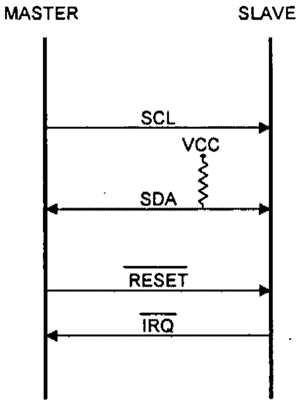


Figure 48: I2C interface connections

The signals shown in the diagram are defined as follows:

- SCL – Serial Clock – This signal is generated by the master and is used to clock serial data from either device. The clock signal provides 2 basic functions. First when set to a logic high level, the SDA line determines start and stop timing, and second it clocks the serial data. Data is clocked out of either device on the falling edge and clocked into the receiving device on the rising edge.
- SDA – Serial Data – This signal is a half duplex serial data line. This signal is open drain where either device may sink to drive the line to a logic low. The pullup resistor is the source for a logic 1 condition.

The SDA line also provides the framing for the transmission of data. See the following waveform diagrams.

- /RESET – Reset (low True) – This signal is driven by the master and received at the slave. When driven to a logic low level, the slave device is held in a reset state. When driven to a logic high level, the slave device is operational.
- /IRQ – Interrupt Request (low true) – This signal is driven from the slave device and received at the master device. Another name for this signal might be "attention", since the slave device uses this line to gain the attention of the master that it is requesting to be serviced.

The I2C interface signaling is too complex to illustrate all of the possibilities in this document. The reader should acquire a copy of "The I2C Bus Specification", Version 2.1, and dated January 2000 from:

http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf

To summarize, the I2C is a half duplex serial protocol with start and stop signaling. It also contains a device address, read/write flag, and acknowledgement field for every 8 bits transferred. The interface requires one device to be the master and driver the SCL (clock) line.
If the slave has information to send the master, the slave must assert the /IRQ line to get the masters attention. The master must then make the transfer.

Typical clock rates found in most products that support this protocol is 100kbps and 400kbps, but the I2C specification specifies the rate can be as high as 3.4Mbps.

There are 2 distinct advantages to this approach. First, there is no tolerance requirement between the master and slaves reference clocks since a clock is provided with the serial bit stream, and second, there is a minimum number of pins required..

One distinct disadvantage is the master must be involved in any transfers from the slave.

6.1.1.3 Universal Asynchronous Receiver Transmitter

The Universal Asynchronous Receiver Transmitter (UART) is a 2 wire full duplex asynchronous interface that allows for low to moderate speed communications with low pin count. This interface has many references such as the "Comm" or "RS232" port on a PC. The UART interface described here follows the data format of these other references but does not include the level translator, keeping the interface a +3VDC logic levels. Although there is many protocols that exist that can be layered on top of the UART communications, the UART itself is simple and strait forward to understand.

The UART interface only defines the physical interface and does not have a transport protocol.

Figure 49 illustrates the UART interconnect. The primary UART interface consists of four wires of which two signals are for communications and two signals for control/status.

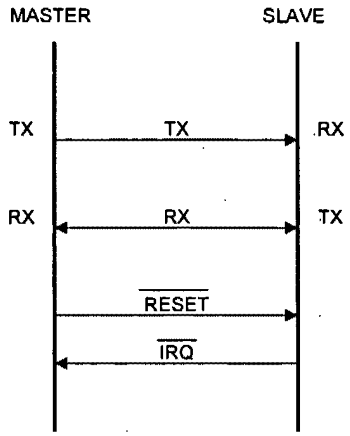
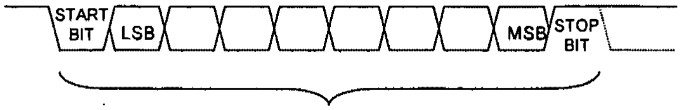


Figure 49: UART interface connections

The signals shown in the diagram are defined as follows:

- TX – Transmit Data – This signal is an asynchronous serial data line that carries a transmit serial data stream from the master to the slave. This signal is defined as transmit since all references are made to the master side.
- RX – Receive Data – This signal is an asynchronous serial data line that carries a transmit serial data stream from the slave to the master. This signal is defined as receive since all references are made to the master side.
- /RESET – Reset (low True) – This signal is driven by the master and received at the slave. When driven to a logic low level, the slave device is held in a reset state. When driven to a logic high level, the slave device is operational.
- /IRQ – Interrupt Request (low true) – This signal is driven from the slave device and received at the master device. Another name for this signal might be "attention", since the slave device uses this line to gain the attention of the master that it is requesting to be serviced. This signal may only be used to wake up the master device if the master is in sleep mode.

Figure 50 shows a basic one-byte transfer using the UART interface. Since the communications interface is asynchronous no clock is needed, only a single serial data line in each direction. The diagram shown can be generated at either end when either device has data to transmit.



1 start bit, 8 data bits, 1 stop bit
(1 byte transfer)

Figure 50: UART Interface waveform diagram

The diagram is in reference to the output of the UART in digital logic format.
Each UART when idle maintains a logic high level on their transmit output. When a byte is to be transmitted, a one bit time logic low is asserted indicating a start condition. The next 8 bit times indicate the data byte being transmitted with the lsb being transmitted first. A stop condition then follows for one bit time for this example. Stop conditions can be extended to multiple positions if what is known as character spacing is required.

After the stop bit the next subsequent byte can be transmitted in the same sequence.

Since each device can transmit data at will, there is no physical master intervention required to retrieve data from the slave.

There are 3 distinct advantages to this approach. First, the master does not need to be physically involved in a slave transmission. Second, various data rates can be achieved to the maximum the master and slave can generate and receive.

One distinct disadvantage is both devices must have a bit rate tolerance within 2.5% (or 25000 PPM). Most microcontroller devices today are capable of achieving this tolerance.

The following sections in yellow should be ignored. A different model will be used including a pseudo-HDLC protocol.

6.1.2 Proxense Chipset Communications Model – RDC configuration

The Proxense chipset communications model is a layered model. The model includes a physical layer that was previously described above, a transport layer that adds and removes framing information to/from the physical transport, an a service layer that interfaces to the applications layer and segments and recombines messages sent to and received from the transport layer.

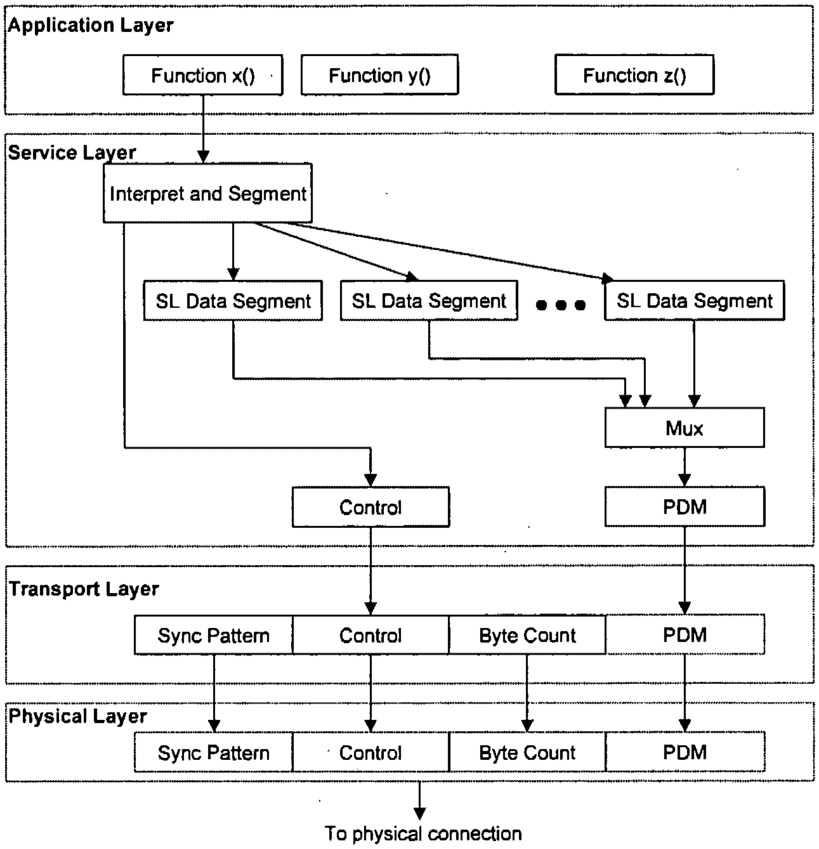


Figure 51: Proxense Communication flow from Application to Physical Layer

Referring to Figure 51, four primary sections are shown. The application layer is either the system application sitting on their server (or other chipset user's computing device) or the application that sits in the RDC for high-level control of the Proxense application. The application layer executes function calls to the service layer.

When a function call occurs, the service layer processes the function call. The service layer interprets the function and if data is present, segments the data into manageable size messages.

Each segmented message becomes a Proxense Data Message (PDM) at the service layer that is then sent to the Transport Layer. Along with the message, the function call may require specific control information to be sent, which the service layer may manipulate and pass on to the transport layer.

The transport layer then takes the PDM and attaches a Synchronization Header, a control information, and a byte count indicating the number of bytes that are in the PDM creating a Proxense Data Packet.

The transport layer then delivers the PDP to the physical layer where the data is transmitted over the physical connection.

As shown in Figure 52, on the receiving end the opposite scenario takes place. The receiving device extracts the data at the physical layer and serially passes the data to the transport layer. The transport layer detects the synchronization pattern and gains PDP synchronization.

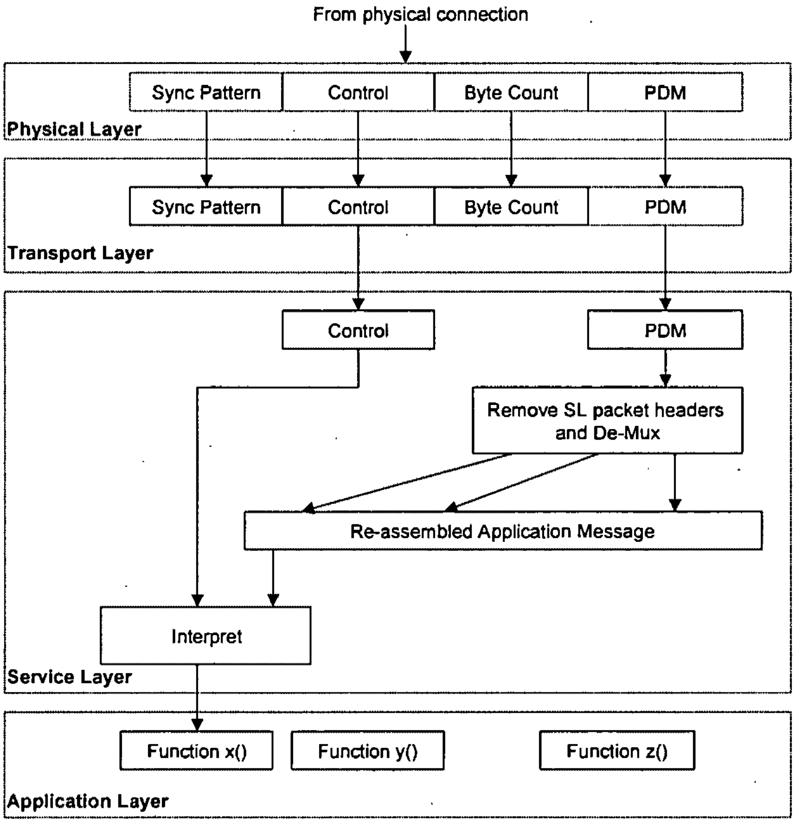


Figure 52: Proxense Communication flow from Physical Layer to Application

Once PDP synchronization has been gained, the control value is retrieved to determine the type of Proxense Data Packet is being received and if any special routing of the data should be considered.

Next the byte count is received which indicates the size of the PDM in the packet. The transport layer then accumulates the specific number of bytes defined by the byte count. Once the transport layer has received the entire PDM contents, the transport layer passes the control information with the PDM contents to the service layer.

The service layer extracts service layer control and byte count information that allows it to determine the message type and application size and reassembles the segments of the message into a complete application message. The application message is then delivered to the application along with any specific control information. The message may arrive at the application layer via an interrupt routine or a function call from the application layer itself.

The transport and service layer protocol structures will now be described in the following sections.

6.1.2.1 Transport Layer Protocol Overview

The Proxense chipset transport layer defines the protocol structure and format of how an external device (referred to in this section as a master) will communicate to the Proxense chipset (referred to as the slave). The transport layer protocol will reside in both the Proxense chipset and system owner's controller.

The protocol specified here is intended for use at the RDC fixed end equipment.

The Proxense Transport Layer protocol structure shall be as defined in this section.

The protocol structure is based on a wired serial communications link. It is expected that the device the Proxense chipset is connected to be in close proximity of the Proxense chip set, so no error detection or correction will be built into the transport layer protocol.

In addition, no source and destination addressing is built into the protocol since the connection shall be point to point.

As shown in Figure 53, the protocol structure at the highest level contains a Proxense Packet Data (PDP).

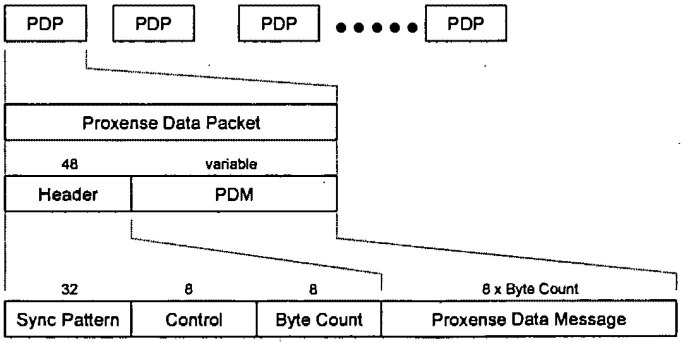


Figure 53: Proxense Transport Layer protocol structure

The PDP is a self-contained packet provides the necessary framing information for the receiving unit to detect the presence of a packet, the type of packet, the length and contents of the packet.

The PDP is broken down into 2 fields, the Header, and the Proxense Data Message (PDM). The header provides specific information for the receiver to synchronize to and determine the physical size of the packet without having to understand the contents of the PDM. The PDP only provides the means for transporting the message from one device to another device.

The PDM provides the means to carry the higher layer messages between devices. The content of the PDM has no meaning to the transport layer.
The header is further broken down into 3 sub-fields. The 3 sub-fields are defined as follows:

- Sync Pattern – The Sync Pattern is a 32 bit (4 byte) unique pattern that allows the receiving device to detect the beginning of a PDP. The synchronization pattern shall be: 0xAA55FF00 (hexadecimal).
- Control – The Control field is an 8 bit (1 byte) field that defines the type of packet being transmitted and if any special routing should be considered. The current values for this field can be found in Table 11.
- Byte Count – The Byte Count field defines the number of bytes contained in the PDM. The Byte Count can range from 0 to 255 decimal.

All data carried in the PDM must be left aligned within the PDM and no filler shall be appended after the data.

6.1.2.1.1 Control Field values

The control field contains information that specifies the PDP and PDM formats and any unique routing requirements. The current values defined for the control field are:

Name	Bits	Definition
PDM Format	2:0	This value defines the format of the data contained within the PDM Value Description 0 Raw Data (default) 1 Numbered Messages 2 – 7 Reserved for future use
RFU	3:5	Reserved for Future Use
Routing Select	6	This value defines where data is routed on the receiving device. Value Description 0 Service Layer 1 Application Layer
Reserved	7	Reserved – this bit shall be set to 0

Note: A field that is "Reserved" must always stay at the value defined. If the field is any value other than the defined value, the receiving device shall discard the entire PDP.

A field that is labeled "RFU" may be ignored and the receiving end will mask these bits to continue normal operation.

Table 11: Control Field definition

PDM Format[2:0] – Proxense Data Message Format – This field defines the format of the contents in the PDM. Currently only two formats exist, but the field has been defined to allow for up to a total of 8 formats.

Raw Data Format (0) - This value indicates that the data from the service layer is packed into the PDM up to a maximum capacity of 255 bytes. No additional information is included in the PDM.

Numbered Messages (1) – This value indicates that the first byte of the PDM will always contain a sequence number. Service layer data is then appended to the sequence number up to the maximum capacity of the PDM (in this case 255-1 bytes are reserved for service layer data). Each transmission of a PDP will increment the sequence number. The byte count in the PDP header shall reflect the total number of bytes in the PDM including the sequence number.

RFU (2 – 7) – These values are currently undefined and are reserved for future use. If a receiving device detects one of these values, the receiving device shall ignore and discard the message.

RFU [3:5] – These 3 bits are reserved for future use and shall be set to a value of 0. All receivers shall ignore this field.

Routing Select [6] - This bit indicates the source and destination of the PDM data. There are only 2 sources/destinations.

Routing Select (0) - A value of zero indicates that the PDM data is to be routed only to the internal service layer and shall never be routed to the application layer. By allowing routing directly to the service layer, specific service layer control and status information can be processed without involvement of the application layer. One example would be link integrity between the two endpoints.

Routing Select (1) - A value of one indicates that the PDM data is to be routed to the application layer. In normal operating mode, this is the default value.

Reserved [7] – This is a reserved field that shall be set to zero. Any device detecting this field with a value other than zero shall ignore and discard all data until the next PDP header is detected.

6.1.2.2 Service Layer Protocol Overview

The Proxense chipset service layer defines the protocol structure between the application layer and the transport layer. This service layer will reside in both the Proxense chipset and the system controller.

The service layer provides several functions.

- Segmentation of application layer messages
- Service Layer formatting
- Segment numbering
- Low level Service Layer to Service Layer primitive communications (i.e. link maintenance, etc.)
- Message synchronization and reassembly
- Presentation of message and control information to the application layer

A basic representation of the above functions is shown in Figure 51and Figure 52.

The segmentation of an application layer message is required, when the application layer message exceeds the length of what the transport is capable of carrying in a single PDP. As described in section 6.1.2.1.1, the PDM may be set for raw format or for numbered messages. The service layer is responsible for providing the sequence number if in numbered message format.

Figure 54 illustrates the partition of a message being sent that is larger than can be carried in a single PDM. The application layer message in this example is 300 bytes. The message is then broken into 2 PDMs where the first PDM carries 255 bytes of the message and the second PDM carries 45 bytes of the message.

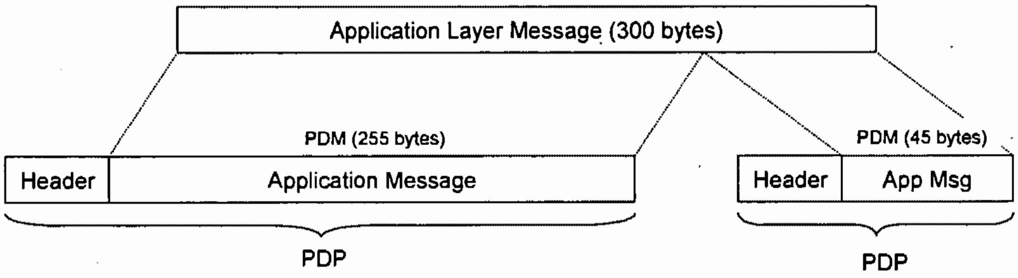


Figure 54: Application Layer Segmentation (Raw Data Format)

The message is then sent across the physical link and reassembled at the receiving end. In Figure 55 the same application is sent, but the PDM format is set to numbered messages indicating that the first byte of each PDM will be an incremental sequence number.

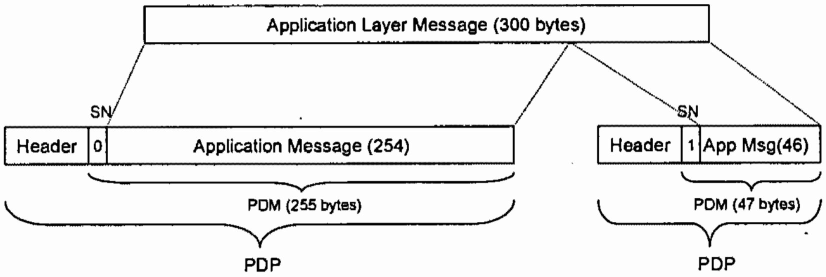


Figure 55: Application Layer Segmentation (Numbered Message Format)

The application message is again segmented, but the sequence number accounts for the first byte of each PDM, so the first PDM can only carry 254 bytes, and the remaining 46 bytes are placed in a second PDM.

In the header, the byte count for the first PDM will be 255 reflecting one byte for the sequence number and 254 bytes for the application message. The header for the second PDM will reflect a byte count of 47 that includes the sequence number and remaining application bytes.

Upon reception of the information by the receiving device, the transport layer passes the PDM with any control information to the service layer. The service layer decodes the control field and then performs the appropriate function on the PDM data.

6.1.3 Proxense Chipset Protocol Handshake

In order for the system controller to interface with the Proxense chipset, a defined set of transfer transactions must exist. The transfer transaction defines how a transaction will take place between the system controller and the Proxense chipset.

Three types of transfer transactions are defined as follows:

- Transmit with no response
- Transmit with acknowledge response
- Transmit with data response

A transmit with no response transaction, is a unidirectional transfer where data is sent from the first device to the second device without expecting a response. A typical example of this would be where the RDC is placed in a mode to transmit detection of a POD in range to the system controller. Upon detection of a POD in range, the Proxense chipset would transmit data to the system controller indicating that a POD was in range along with the POD's identifying information.

A transmit with acknowledgement response transaction, is a bi-directional transfer where a command or data is sent from the first device to the second device, and the second device transmits an acknowledgement message back to the first device indicating it received the message. A typical example of would be the system controller commanding the RDC to go to channel 2. Upon detection of this command by the RDC, the RDC would acknowledge that it received the command, and then would switch to channel 2.

A transmit with data response transaction, is a bi-directional transfer where a command or query of information is requested by the first device and the second device responds with the relative data. A typical example of this would be the system controller requesting the RDC to return all POD IDs that are in association mode. Upon detection of the request, the RDC would gather all associated POD information and return the information to the system controller.

6.1.4 Proxense Software Architecture

This section will describe the overall Proxense software architecture and how it exists in the system controller and in the Proxense Chipset. The Proxense TruProx chipset provides a layered approach.

By providing a Proxénse Stack, the system operator need only attach a physical layer driver and add the necessary software to interact with the Proxense service layer, i.e. function calls.

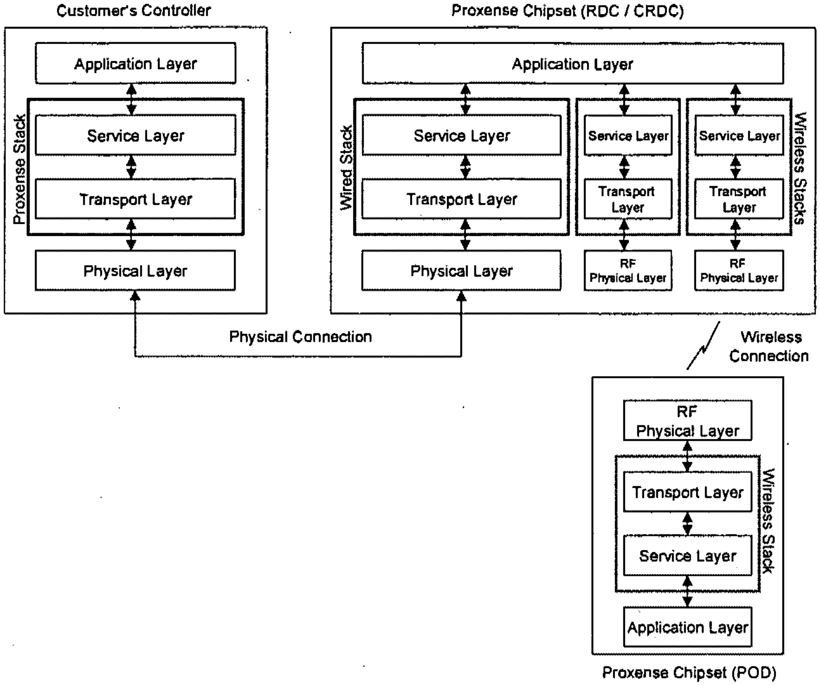


Figure 56 Poxense High Level Layered Model

Figure 56 illustrates the layered model that is envisioned for the Proxense chipsets and protocols. Referring to the left-hand side of the diagram, the system controller contains the system application layer. This allows the operator to define functionality of the application layer per customer or property application requirements. Proxense can then provide guidance for operation and features related to the Proxense system.

The Proxense stack connects into the application layer through a set of function calls that will be later described. The Proxense stack includes the service and transport layers that may have specific memory requirements on the system controller.

The Proxense Stack transport layer interconnects to the system physical layer driver, which in turn drives the communications link between the system controller and the Proxense chipset. At this time a simple UART operating at 3VDC digital logic level is being considered for this interface.

The system controller is then connected through the physical connection to the Proxense chipset, where the equivalent physical, transport, and service layer resides. The Proxense chipset application layer controls the Proxense proprietary system and provides interconnect between the system controller and the Proxense POD.

The layers between the Proxense chipset RDC and the Proxense POD also follow a similar approach but implements a wireless stack allowing proprietary Proxense device to communicate.

Although not shown, the second RF physical layer located in the Proxense chipset RDC block may connect to a second Proxense chipset RDC.

6.1.5 Proxense Chipset Application Functions in an RDC

The Proxense chipset provides several major functions in its application layer. As shown in the Proxense RDC configuration in Figure 57 the application layer sits between the wired and wireless interfaces, providing the functionality as previously explained in this document.

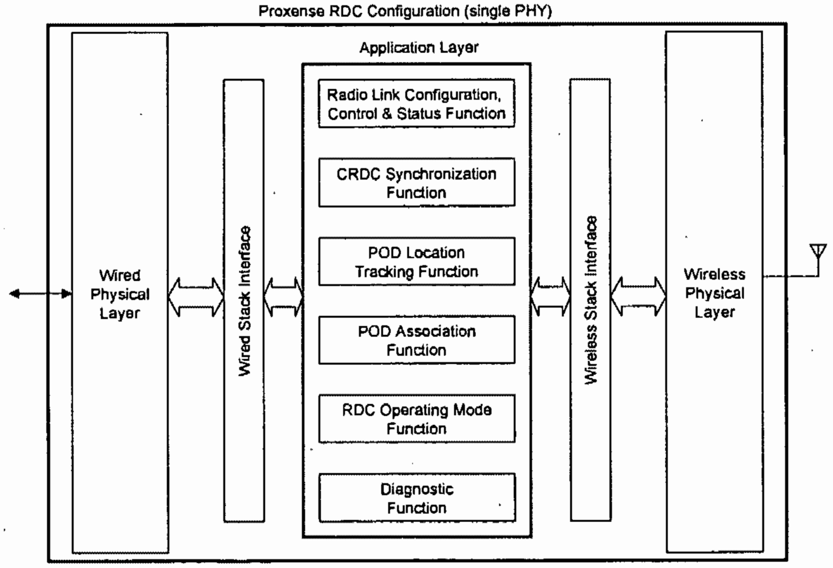

Proxense RDC Configuration (single PHY)
Application Layer
Wired Physical Layer
Wired Stack Interface
Radio Link Configuration, Control & Status Function
CRDC Synchronization Function
POD Location Tracking Function
POD Association Function
RDC Operating Mode Function
Diagnostic Function
Wireless Stack Interface
Wireless Physical Layer

Figure 57: Proxense chipset application functional blocks in an RDC

The above diagram illustrates an RDC with a single wireless physical interface. To reduce overall confusion, the RDC with a single wireless PHY is much easier to explain. An RDC with a dual wireless PHY will effectively duplicate the wireless stack and physical layers.

Each of the functions shown, provide a necessary function for the RDC to operate and provide communications to the system controller and the POD.

The radio link configuration, control and status function deals with how the RF Protocol is configured and provides the link maintenance to establish and maintain wireless connections with the POD.

The CRDC synchronization function recovers and maintains timing received from a CRDC. This information is used to align the RDC in the system.

The POD location tracking function recovers POD related information including the POD ID, signal quality metrics, and a timestamp that is derived from the CRDC synchronization function. After collecting the information related to a POD, the information is only temporarily stored and passed to the service layer in the system controller. This function has a limited storage area and will only be capable of maintaining the PODs information for a limited time.
This function also provides the switch channel response to a POD location tracking response message when the application layer wants to associate with the POD.

The POD association function has several sub-functions included by not shown. The POD association function provides the means to establish and maintain a link between the RDC and POD. In addition, the association function contains the sub-functions required to perform authentication of the POD and maintain a secure link. Any secure data transactions between the RDC and POD are executed through this function.
This function and current memory requirements only provide for a single POD association.

The RDC operating mode function is a static type function that is configured during initial installation. This function defines the network configuration and environment the RDC is integrated into.

The Diagnostic function allows the RDC to perform local diagnostic tests to determine if the device is operating properly. This function is also used during installation to tune RF related parameters for the environment.

6.1.6 Proxense Service Layer Functions in the System Controller

As previous described the system controller will contain a Proxense stack. The stack serves as the interface to the RDC automating many of the functions and gathering information for the system application layer.

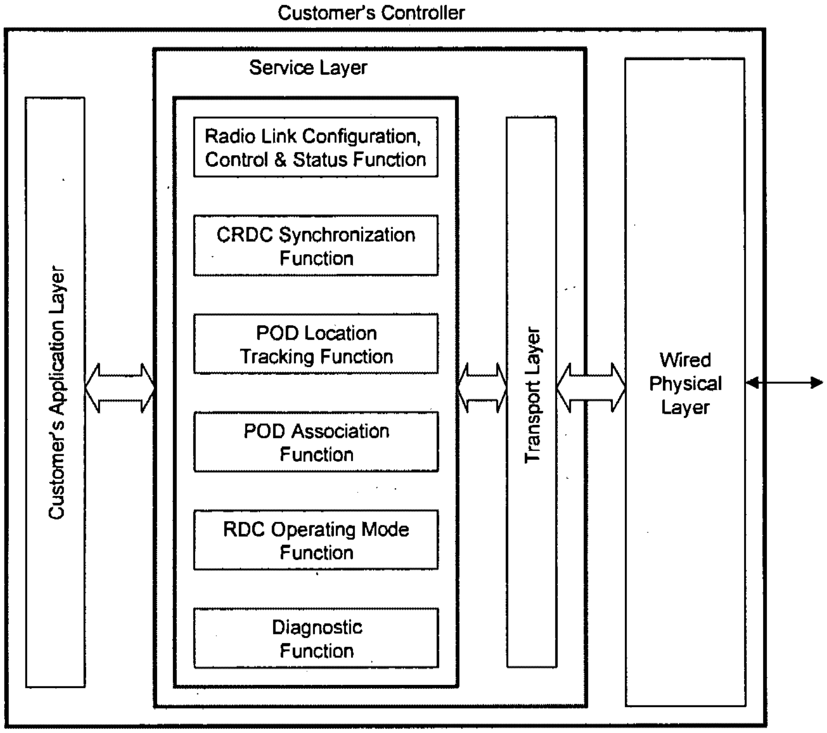

Customer's Controller
Service Layer
Radio Link Configuration, Control & Status Function
CRDC Synchronization Function
POD Location Tracking Function
POD Association Function
RDC Operating Mode Function
Diagnostic Function
Customer's Application Layer
Transport Layer
Wired Physical Layer

Figure 58: System controller showing Proxense Service Layer functions

As shown in Figure 58, functions that were previously described in the Proxense chipset application layer also reside in the Proxense stack on the controller. Although they have the same name, the functions themselves are different in nature. Most of the functions in the Proxense stack are streamlined for efficiency since the controller is not burdened with the requirement to service the wireless interfaces. That is managed internally within the Proxense chipset.

The Radio link configuration, control and status function provides configuration information to the RDC and is periodically called upon to change channel assignments or monitor radio related status.

The CRDC Synchronization Function is again mostly provides configuration information related to the network. It also has the ability to poll or receive current timing information related to the RDCs synchronization to the CRDC.

The POD location tracking function is a more complex function for the service layer. This function must maintain a database containing recently active POD IDs within the RDC cell. The database will vary in length based on the traffic an RDC detects.

In addition, this function must also frequently communicate with the RDC to retrieve the most recent POD information.
This is most likely the most active function in the service layer.

The POD association function is another more complex function requiring frequent interaction with both the RDC and a POD while initiating, maintaining, and terminating association through the wireless link. Based on an application layer request, this function will interact with the RDC to initiate the link and will exchange information necessary to maintain the link. It must also monitor and log signal quality information that is made available to the application layer.
This function also differs from the Proxense chipset application layer in that it does not perform any of the Proxense proprietary security measures with the POD. It only has the capability to request the RDC to establish a secure link and report status of an authentication.
In addition to initiating and maintaining association, this function is also responsible for the transport of data to and from the application layer, the RDC, and the PODs internal database.

The RDC operating mode function mostly provides configuration information to the RDC during installment of or reconfiguration of the RDC.

The Diagnostic function mostly provides commands to and receives status from the RDC related to specific health of the RDC.

6.1.7 Proxense Stack Function Calls

To ease the implementation into an application, the Proxense stack was developed. The Proxense stack offers a set of easy to implement C function calls. The stack handles the interpretation and presentation of information to the application layer as well as all interactions with the Proxense chipset.

The functions are broken down into 3 groups as follows:

- Proxense Chipset Operating Mode Configuration
- Proxense Chipset Radio Link Configuration
- POD Communications

The syntax for all function calls to the Proxense stack will be written in the ANSI-C programming language.

6.1.7.1 Bit and Byte Ordering

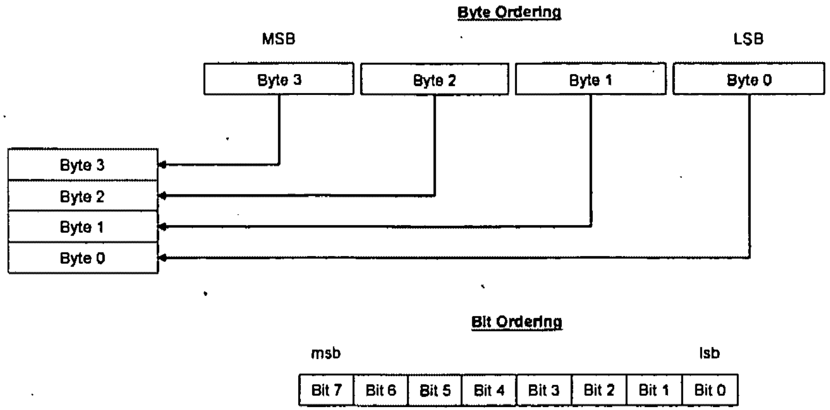


Figure 59: Byte and Bit ordering diagram

The Most Significant Byte (MSB) shall be the leftmost or first byte, and the Least Significant Byte (LSB) shall be the rightmost or last byte.

The most significant bit (msb) shall be the leftmost or first bit, and the least significant bit (lsb) shall be the rightmost or last bit.

6.1.7.2 RDC Operating Mode Configuration

The operating mode configuration functions define the operating conditions for the Proxense chipset.

6.1.7.2.1 Check wireline connection (check_connection)

The check connection function performs a communications test between the Proxense stack on the system controller and the Proxense chipset to verify the two devices are communicating. This function returns a value that indicates pass or fail.

Syntax: char check_connection(void)

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.2.2 Power up Proxense Chip Set (power_up_pcs)

The power_up_pcs function sends a command to the Proxense chipset to power up all internal circuits. This places the chipset into an idle operating condition.
This function returns a value that indicates pass or fail.

Syntax: char power_up_pcs(void);

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.2.3 Power down Proxense Chip Set (power_down_pcs)

The power_down_pcs function sends a command to the Proxense chipset to power down all internal circuits except for those required enabling wire line communications to the device. This places the chipset into an off operating condition.
This function returns a value that indicates pass or fail.

Syntax: char power_down_pcs (void)

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.2.4 Reset Proxense Chip Set (reset_pcs)

The reset_pcs function instructs the Proxense chipset to perform a hard reset causing all circuits to be reinitialized. The Proxense chipset will respond indicating that the command has been received prior to performing a hard reset.

Syntax: char reset_pcs (void);

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.2.5 Set Proxense Chip Set operating mode (set_pcs_op_mode)

The set_pcs_op_mode function defines the RF physical interface and operation of the local Proxense stack on the controller and the Proxense chipset that an RDC will function as. There are 4 basic operating modes:

- Not configured
- Single Cell RDC
- Coordinated Multi-cell RDC
- Coordinated Multi-cell CRDC coordinator

The single cell operating mode configures the stack and Proxense chipset to operate as a single cell RDC creating its own beacon and operating on a single frequency. In this mode the Proxense chipset does not use the CRDC for synchronization, but uses internal timing to generate its beacon. In the Coordinated Multi-cell RDC mode, the Proxense chipset uses the both RF physical interfaces with one of the RF Physical interfaces for synchronization, monitoring of POD location tracking responses, and commanding a POD to switch channels. The second RF physical interface is used for association with the POD.

This function returns a value that should reflect the mode selected, or -1 if the unit failed to change modes.

Syntax: char set_pcs_op_mode (uchar phy_val, uchar mode_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

mode_val	Meaning
0	None
1	Single Cell RDC
2	Coordinated Multi-cell RDC
3	Coordinated Multi-cell CRDC
4-127	RFU

Returned Value:

Value	Meaning
-1	Fail
0	None
1	Single Cell RDC
2	Coordinated Multi-cell RDC
3	Coordinated Multi-cell CRDC
4-127	RFU

6.1.7.2.6 Get Proxense Chip Set operating mode (get_pcs_op_mode)

The get_pcs_op_mode function instructs the service layer to get the current operating mode from the Proxense chipset as defined in section 6.1.7.2.5.

This function returns a value that should reflect the operating mode the Proxense chipset is currently operating, or -1 if the unit failed to change modes.

Syntax: char get_pcs_op_mode (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

Value	Meaning
-1	Fail
0	No operating mode specified
1	Single Cell RDC
2	Coordinated Multi-cell RDC
3	Coordinated Multi-cell CRDC
4-127	RFU

6.1.7.2.7 Operating Mode Enable (op_mode_en)

The op_mode_en function enables the operation mode for a specific RF physical interface. The mode of operation is defined using the set_pcs_op_mode function defined in section 6.1.7.2.5.

This function returns a value that indicates pass or fail.

Syntax: char op_mode_en (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.2.8 Operating Mode Disable (op_mode_dis)

The op_mode_dis function immediately disables the current operation mode for a specific RF physical interface. The mode of operation is defined using the set_pcs_op_mode function defined in section 6.1.7.2.5. The operation on the specific RD physical interface immediately terminates and the RF physical interface remains in an idle mode.

This function returns a value that indicates pass or fail.

Syntax: char op_mode_dis (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3 Proxense Chipset Radio Link Configuration

The radio link configuration functions define the parameters the Proxense chipset will use for operating and maintaining the RF physical links from CRDCs and to/from PODS.

6.1.7.3.1 Set Proxense chip set ID value (set_pcs_id_val)

The set_pcs_id_val function defines the ID value for a Proxense chipset on a specific RF physical interface. This value is used to indicate that this device is transmitting data or detected to determine if received data is for this device.

This function returns a pass or fail indication.

Syntax: char set_pcs_id_val (uchar phy_val, uint tx_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

tx_id	Meaning
0 - 65535	Value of ID for this Proxense chipset

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.2 Get Proxense chip set ID value (get_pcs_id_val)

The get_pcs_id_val function instructs the service layer to get the current ID used for a Proxense chipset on a specific RF physical interface.

Syntax: uint set_pcs_id_val (uchar phy_val, uint tx_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

tx_id	Meaning
0 - 65535	Value of ID for this Proxense chipset

6.1.7.3.3 Set Proxense chipset receive sync ID value (set_psc_rx_sync_id_val)

The set_pcs_rx_sync_id_val function defines if receive synchronization is enabled and the ID value that a Proxense chipset will listen and synchronize to on a specific RF physical channel. This value is used to gain synchronization of an RDC or CRDC in a coordinated multi-cell configuration.

This function returns a pass or fail indication.

Syntax:

```
        char set_psc_rx_sync_id_val (uchar phy_val, uchar sync_en,uint rx_sync_id)
```

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

sync_en	Meaning
0	Do not synchronize
1	Synchronize to over-the-air Receive Sync ID

rx_sync_id	Meaning
0 - 65535	Value of CRDC ID that this RDC will synchronize to

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.4 Get Proxense chipset receive sync ID value (get_psc_rx_sync_id_val)

The get_pcs_rx_sync_id_val function instructs the service layer to get the current ID value from the Proxense chipset that a specific RF physical interface will synchronize to.

Syntax: * uchar get_psc_rx_sync_id_val (uchar phy_val, * rx_sync_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

An array stored in the rx_sync_array.

rx_sync_array	
sync_en	Synchronization Enable 0 = do not synchronize 1 = synchronize to over-the-air Receive Sync ID
rx_sync_id(MSB)	Receive Sync ID (MSB)
rx_sync_id(LSB)	Receive Sync ID (LSB)

6.1.7.3.5 Set Beacon channel availability flags (set_beacon_ch_avail_flags)

The set_beacon_ch_avail_flags function defines the contents of the channel_availability field in a beacon transmission for a specific RF physical interface. This is an integer value with each bit representing a channel number. This information is broadcast to inform receiving devices what channels are occupied by beacons.

This function returns a pass or fail indication.

Syntax:

char set_beacon_ch_avail_flags (uchar phy_val, uint avail_flags)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

avail_flags	Meaning
Bit 15	Channel 15 Flag
Bit 14	Channel 14 Flag
Bit 13	Channel 13 Flag
Bit 12	Channel 12 Flag
Bit 11	Channel 11 Flag
Bit 10	Channel 10 Flag
Bit 9	Channel 9 Flag
Bit 8	Channel 8 Flag
Bit 7	Channel 7 Flag
Bit 6	Channel 6 Flag
Bit 5	Channel 5 Flag
Bit 4	Channel 4 Flag
Bit 3	Channel 3 Flag
Bit 2	Channel 2 Flag
Bit 1	Channel 1 Flag
Bit 0	Channel 0 Flag

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.6 Get Beacon channel availability flags (get_beacon_ch_avail_flags)

The get_beacon_ch_avail_flags function instructs the service layer to get the current beacon channel availability flags parameter from the Proxense chipset as defined in section 6.1.7.3.5 for a specific RF physical interface.

This function returns a value that should reflect the current beacon channel availability values that the Proxense chipset of a specific RF physical interface is currently transmitting.

Syntax:

```
uint get_crdc_ch_avail_flags (uchar phy_val)
```

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

avail_flags	Meaning
Bit 15	Channel 15 Flag
Bit 14	Channel 14 Flag
Bit 13	Channel 13 Flag
Bit 12	Channel 12 Flag
Bit 11	Channel 11 Flag
Bit 10	Channel 10 Flag
Bit 9	Channel 9 Flag
Bit 8	Channel 8 Flag
Bit 7	Channel 7 Flag
Bit 6	Channel 6 Flag
Bit 5	Channel 5 Flag
Bit 4	Channel 4 Flag
Bit 3	Channel 3 Flag
Bit 2	Channel 2 Flag
Bit 1	Channel 1 Flag
Bit 0	Channel 0 Flag

6.1.7.3.7 Set Radio Frequency operating channel (set_rf_op_ch)

The set_rf_op_ch function defines the RF channel a Proxense chipset will normally operate on for a specific physical interface.

This function returns a pass or fail indication.

Syntax: char set_rf_op_ch (uchar phy_val, uchar rf_ch)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

rf_ch	Meaning
0 - 15	Defines the channel number a Proxense chipset will operate on

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.8 Get Radio Frequency operating channel (get_rf_op_ch)

The get_rf_op_ch function instructs the service layer to get the RF channel a Proxense chipset will normally operate on for a specific physical interface.

Syntax: uchar get_rf_op_ch (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

rf_ch	Meaning
0 - 15	Defines the channel number a Proxense chipset will operate on

6.1.7.3.9 Set Timeslot length value (set_ts_len_val)

The set_ts_len_val function defines the number of symbols allocated to a timeslot. This value is a constant in the system and is used by the CRDC, RDC, and POD to determine the timeslot length of the Time Division Multiple Access (TDMA) system. This value defines the internal timing of the timeslot period and the value that will be transmitted in the beacon.

Note: If a Proxense chipset has set the sync_en bit in the set_psc_rx_sync_id_val function in section 6.1.7.3.3, if a timeslot length value is contained in the beacon, the received timeslot information will over-ride this value.

The default value for this value is 306 symbols.

Syntax: char set_ts_len (uint ts_len)

ts_len	Meaning
0 - 400	Defines the number of symbols per timeslot

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.10 Get Timeslot length value (get_ts_len_val)

The get_ts_len_val function instructs the service layer to get the current timeslot length parameter from the Proxense chipset as defined in section 6.1.7.3.9. If a Proxense chipset is configured as a coordinated multi-cell CRDC or single cell RDC, the value returned is the value programmed into the Proxense chipset unless the sync_en bit in the set_psc_rx_sync_id_val is set, in which case the returned value is the over-the-air received value.

Syntax: uint get_ts_len_val (void)

Returned Value:

ts_len	Meaning
0 - 400	Defines the number of symbols per timeslot

6.1.7.3.11 Set Superframe length value (set_sf_len_val)

The set_sf_len_val function defines the number of timeslots allocated to a superframe. This value is a constant in the system and is used by the CRDC, RDC, and POD to determine the superframe length of the Time Division Multiple Access (TDMA) system. This value defines the internal timing of the superframe period and the value that will be transmitted in the beacon. The value is computed as 2^{sf_len}.

Note: If a Proxense chipset has set the sync_en bit in the set_psc_rx_sync_id_val function in section 6.1.7.3.3, if a superframe length value is contained in the beacon, the received superframe information will over-ride this value.

The default value is 4 defining 2^4 or 16 timeslots.

This function returns a pass or fail indication.

Syntax: char set_sf_len_val (uchar sf_len)

sf_len	Meaning
0 – 15	Defines the number of timeslots in a superframe. The number of timeslots is computed as 2^{sf_len}.

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.12 Get Superframe length value (get_sf_len_val)

The get_sf_len_val function instructs the service layer to get the current superframe length parameter from the Proxense chipset as defined in section 6.1.7.3.12. If a Proxense chipset is configured as a coordinated multi-cell CRDC or single cell RDC, the value returned is the value programmed into the Proxense chipset unless the sync_en bit in the set_psc_rx_sync_id_val is set, in which case the returned value is the over-the-air received value.

Syntax: uchar get_sf_len_val (void)

Returned Value:

sf_len	Meaning
0 - 15	Defines the number of timeslots in a superframe. The number of timeslots is computed as 2^{sf_len}.

6.1.7.3.13 Set C-Superframe length value (set_csf_len_val)

The set_csf_len_val function defines the number of superframes allocated to a C-superframe. This value is a constant in the system and is used by the CRDC, RDC, and POD to determine the C-superframe length of the Time Division Multiple Access (TDMA) system. This value defines the internal timing of the C-superframe period and the value that will be transmitted in the beacon. The value is computed as 2^{csf_len}. The value is computed as 2^{csf_len}.

Note: If a Proxense chipset has set the sync_en bit in the set_psc_rx_sync_id_val function in section 6.1.7.3.3, if a C-superframe length value is contained in the beacon, the received C-superframe information will over-ride this value.

The default value is 2^5 defining 32 superframes.

This function returns a pass or fail indication.

Syntax: char set_csf_len_val (uchar csf_len)

csf_len	Meaning
0 – 15	Defines the number of superframes in a C-superframe. The number of C-superframes is computed as 2^{csf_len}.

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.14 Get C-Superframe length value (get_csf_len_val)

The get_csf_len_val function instructs the service layer to get the current C-superframe length parameter from the Proxense chipset as defined in section 6.1.7.3.13. If a Proxense chipset is configured as a coordinated multi-cell CRDC or single cell RDC, the value returned is the value programmed into the Proxense chipset unless the sync_en bit in the set_psc_rx_sync_id_val is set, in which case the returned value is the over-the-air received value.

Syntax: uchar get_csf_len_val (void)

Returned Value:

csf_len	Meaning
0 - 15	Defines the number of superframes in a C-superframe. The number of superframes is computed as 2^{csf_len}.

6.1.7.3.15 Set POD mask value (set_pod_msk_val)

The set_pod_msk_val function defines the mask used over the superframe and timeslot counts and service providers POD ID to determine the superframe and timeslot the POD is active on in the framing structure. See section 5.2.1.2.1.6 for details.
This value is transmitted in a beacon.

Note: If a Proxense chipset has set the sync_en bit in the set_psc_rx_sync_id_val function in section 6.1.7.3.3, if a set_pod_msk value is contained in the beacon, the received value will over-ride this value.

This function returns a pass or fail indication.

Syntax: char set_pod_msk_val (uint pod_msk)

Name	pod_msk (bits)	Meaning
0	Bits 15:0	Always 0
sf_mask	14: 2^{sf_len}	Superframe Mask
ts_mask	2^{sf_len}-1: 0	Timeslot Mask

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.16 Get POD mask value (get_pod_msk_val)

The get_pod_msk_val function instructs the service layer to get the current POD superframe and timeslot mask parameters from the Proxense chipset as defined in section 6.1.7.3.15, unless the sync_en bit in the set_psc_rx_sync_id_val is set, in which case the returned value is the over-the-air received value.

Syntax: uint get_pod_msk_val (void)

Returned Value:

Name	pod_msk (bits)	Meaning
0	Bits 15:0	Always 0
sf_mask	14: 2^{sf_len}	Superframe Mask
ts_mask	2^{sf_len}-1: 0	Timeslot Mask

6.1.7.3.17 Set Site ID value (set_site_id_val)

The set_site_id_val function defines the Site ID value that is transmitted in a beacon on a specific RF physical interface.

Note: If a Proxense chipset has set the en_ota_site_id flag in this function the over-the-air site id received will be used.

This function returns a pass or fail indication.

Syntax: char set_site_id_val (uchar phy_val, uchar en_ota_site_id, uint site_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

en_ota_site_id	Meaning
0	Use site id in site_id field attached
1	User over-the-air site_id

site_id	Meaning
0 - 65535	Value of Site ID that is transmitted.

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.18 Get Site ID value (get_site_id_val)

The get_site_id_val function instructs the service layer to get the current Enable over-the-air site ID flag and Site ID value from the Proxense chipset as defined in section 6.1.7.3.17, unless the en_ota_site_id is set, in which case the returned value is the over-the-air received value.

Syntax: * uchar get_site_id_val (uchar phy_val, char * site_id_info)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

An array stored in the site_id_info array.

rx_sync_array	
en_ota_site_id	Enable over-the-air site id 0 = use programmed site_id value 1 = use over-the-air site_id value
site_id(MSB)	Site ID (MSB)
site_id(LSB)	Site ID (LSB)

6.1.7.3.19 Set Timeslot select value (set_ts_sel_val)

The set_ts_sel_val function defines the value to output in the Proxense Network field during a beacon transmission on a specific RF physical interface. The value determines how RDCs and PODs access the timeslot structure. Refer to section 5.3.1.1.4 for further information on the operation of the value. This function is only used in a coordinated multi-cell CRDC or as a single cell RDC.

This function returns a pass or fail indication.

Syntax: char set_ts_sel_val (uchar phy_val, uchar ts_sel_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

ts_sel_val	Meaning
0	No timeslot assignment
1	802.15.4. timeslot assignment
2	POD uses even timeslots / RDC uses odd timeslots
3	RDC uses even timeslots / POD uses odd timeslots

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.20 Get Timeslot select value (get_ts_sel_val)

The get_ts_sel_val function instructs the service layer to get the current timeslot select value from the Proxense chipset on a specific RF physical interface as defined in section 6.1.7.3.19. If a Proxense chipset is configured as a coordinated multi-cell CRDC or single cell RDC, the value returned is the value programmed into the Proxense chipset. If a Proxense chipset is configured as a coordinated multi-cell RDC, the value returned is the value received from the CRDC.

Syntax: uchar get_ts_sel_val (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

ts_sel_val	Meaning
0	No timeslot assignment
1	802.15.4. timeslot assignment
2	POD uses even timeslots / RDC uses odd timeslots
3	RDC uses even timeslots / POD uses odd timeslots

6.1.7.3.21 Set assigned channel flags (set_asgnd_ch_flags)

The set_asgnd_ch_flags function defines the channels a CRDC or RDC on a specific RF physical interface can monitor and communicate to pods on. This is an integer value with each bit representing a channel number. This value may be altered to coordinate channel reutilization in a Proxense network.

This function returns a pass or fail indication.

Syntax:

char set_asgnd_ch_flags (uchar phy_val, uint asgnd_ch_flags)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

asgnd_ch_flags	Meaning
Bit 15	Channel 15 Flag
Bit 14	Channel 14 Flag
Bit 13	Channel 13 Flag
Bit 12	Channel 12 Flag
Bit 11	Channel 11 Flag
Bit 10	Channel 10 Flag
Bit 9	Channel 9 Flag
Bit 8	Channel 8 Flag
Bit 7	Channel 7 Flag
Bit 6	Channel 6 Flag
Bit 5	Channel 5 Flag
Bit 4	Channel 4 Flag
Bit 3	Channel 3 Flag
Bit 2	Channel 2 Flag
Bit 1	Channel 1 Flag
Bit 0	Channel 0 Flag

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.22 Get assigned channel flags (get_asgnd_ch_flags)

The get_asgnd_ch_flags function instructs the service layer to get the currently assigned channel flags parameter from the Proxense chipset as defined in section 6.1.7.3.21.

This function returns a value that should reflect the current channel assignment the Proxense chipset is using to monitor and communicate to PODs on.

Syntax:

uint get_asgnd_ch_flags (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

asgnd_ch_flags	Meaning
Bit 15	Channel 15 Flag
Bit 14	Channel 14 Flag
Bit 13	Channel 13 Flag
Bit 12	Channel 12 Flag
Bit 11	Channel 11 Flag
Bit 10	Channel 10 Flag
Bit 9	Channel 9 Flag
Bit 8	Channel 8 Flag
Bit 7	Channel 7 Flag
Bit 6	Channel 6 Flag
Bit 5	Channel 5 Flag
Bit 4	Channel 4 Flag
Bit 3	Channel 3 Flag
Bit 2	Channel 2 Flag
Bit 1	Channel 1 Flag
Bit 0	Channel 0 Flag

6.1.7.3.23 Set RDC power level value (set_rdc_pwr_lvl_val)

The set_rdc_pwr_lvl_val function defines the power level value that an RDC or CRDC shall use for transmit power on a specific RF physical interface.

This function returns a pass or fail indication.

Syntax: char set_rdc_pwr_lvl_val (uchar phy_val, char pwr_lvl_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pwr_lvl_val	Meaning
-127 to +127	Value of power level an RDC or CRDC shall transmit at.

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.24 Get RDC power level value (get_rdc_pwr_lvl_val)

The get_rdc_pwr_lvl_val function instructs the service layer to get the current power level value from the Proxense chipset on a specific RF physical interface as defined in section 6.1.7.3.23.

This function returns a value that should reflect the current value the Proxense chipset is operating at. The Proxense chipset may operate at different power levels depending on the mode of operation.

Syntax: char get_rdc_pwr_lvl_val (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

pwr_lvl_val	Meaning
-127 to +127	Value of power level an RDC or CRDC is transmitting at.

6.1.7.3.25 Send change POD power level command (snd_chng_pod_pwr_lvl_cmd)

The snd_chng_pod_pwr_lvl_cmd function defines to the Proxense chipset to send the power level value on a specific RF physical interface to an associated POD to adjust its transmit power level.

This function returns a pass or fail indication.

Syntax: char snd_chng_pod_pwr_lvl_cmd (uchar phy_val, uint pod_id, char pwr_lvl_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID to change power level on

pwr_lvl_val	Meaning
-127 to +127	Value of power level an associated POD shall transmit at.

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.26 Send get POD power level command (snd_get_pod_pwr_lvl_cmd)

The snd_get_pod_pwr_lvl_cmd function instructs the service layer to send a command on a specific RF physical interface to the POD to get the current power level value from the POD. The Proxense chipset will query the associated POD for its power level via the wireless connection.

This function returns a value that should reflect the current value the POD is operating at. The Proxense chipset in the POD may operate at different power levels depending on the mode of operation.

Syntax: char get_pod_pwr_lvl_val (uchar phy_val, uint pod_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID to change power level on

Returned Value:

pwr_lvl_val	Meaning
-127 to +127	Value of power level an associated POD is transmitting at.

6.1.7.3.27 RDC Scan channels (rdc_scan_chans)

The rdc_scan_chans function instructs the Proxense chipset to scan a specified number of channels on a specific physical interface for a specified period of time and defined by the set_asgnd_ch_flags function is section 6.1.7.3.21. The Proxense chipset will scan all of the assigned channels and will return a table indicating the status of each channel.

Syntax:

- rdc_scan_chans (char phy_intfc, uchar num_ts, char * rx_scan_table)

phy_intfc	Meaning
0	RF PHY 0
1	RF PHY 1

num_ts	Meaning
0 to 255	Defines number of timeslots to scan for on a channel

Returned Value: A table that is stored in the address specified by rx_scan_table.

The scan table is of variable length and indicates all devices found on the channels being scanned.

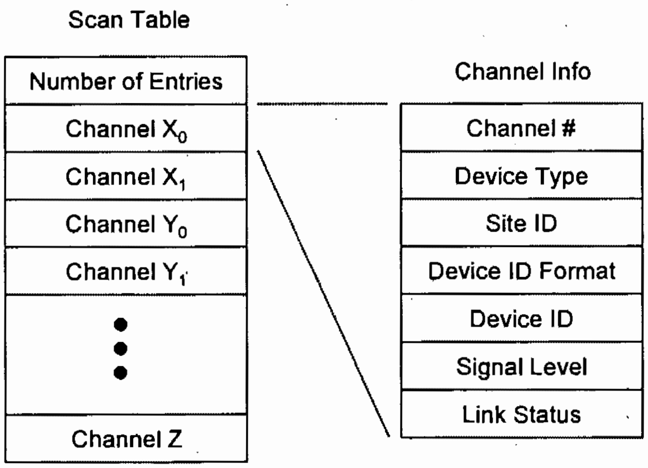


Figure 60: Scan table diagram

As shown in **Error! Reference source not found.**, the scan table will include a one byte header indicating the number of channel information blocks attached. Although in most cases a channel will only contain on information block, there are conditions where multiple devices can be operating on a single frequency with cell overlap and the devices cannot see each other, but the device performing the scan can see both. For this reason, the scan table will include all information found on a channel when scanning.

The channel information block will contain key parameters related to what was found on the channel. Each field in the channel information block is defined below.

6.1.7.3.27.1 Channel Number (type char)

The channel number defines the channel that the following information is reported on. The channel number is defined as follows:

chan_num	Meaning
0 to 15	Channel number info is relative to

6.1.7.3.27.2 Device Type (type char)

The device type defines the type of device found on the channel. The following table will define the device types:

dev_typ	Meaning
0	No device found, only energy level is included.
1	Proxense CRDC detected.
2	Proxense RDC detected.
3	Proxense POD detected.
4	Unknown 802.15.4 device detected.
5-255	RFU

6.1.7.3.27.3 Site ID (type uint)

The site id defines the site id detected on this channel for this device. The site id is only valid for CRDC and RDC devices. This field will be set to 0 if the site id is unknown.

site_id	Meaning
0 to 65535	Site ID value.

6.1.7.3.27.4 Device ID Format (type uchar)

The device id format defines the if the device address received is a 16 bit device ID or a 64 bit device ID. This field will default to a short address if the device address is unknown.

dev_id_fmt	Meaning
0	Short Address (16 bits)
1	Long Address (64 bits)

6.1.7.3.27.5 Device ID (type uchar)

The device id defines the device id detected on this channel for this device. This field will be set to 0 if the device id is unknown. This field will be either a 2 byte field or 8 byte field based on the device ID format field.

dev_id	Meaning
0 to 2^{64}	Device ID value.

6.1.7.3.27.6 Signal level (type char)

The signal level defines the current signal level detected while scanning this channel. If a device is found, it reports the signal level during the reception of the devices transmit packet. If no device was found, it defines the average energy detected on the channel.

sig_lvl	Meaning
-127 to +127	Channel signal level.

6.1.7.3.27.7 Link Status (type uchar)

The link status defines the current conditions detected while receiving data from a device. This value determines if errors are found in the receive data. This field will be set to 0 if no device was found.

lnk_stat	Meaning
0	No errors detected
1	Errored packet received.

6.1.7.3.28 Clear scan status (clr_scn_stat)

The clr_scn_stat function sends a command to the Proxense chipset to clear the scan status for a specific RF physical interface. The status may be status that was automatically collected, or may be status that was requested as defined in section 6.1.7.3.27 above.

This function returns a value that indicates pass or fail.

Syntax: char clr_scn_stat (uchar phy_val)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

Value	Meaning
-1	fail
0	pass

6.1.7.3.29 Get scan status (get_scn_stat)

The get_scn_stat function returns a scan table on a specific RF physical interface. The returned table will be the scan status of all channels that occurred either by the rdc_scan_chans function defined in section 6.1.7.3.27 or by the background automated process.
The returned table will be as defined in section 6.1.7.3.27.

Syntax: * uchar get_scn_stat (char phy_intfc, char * rx_scan_table)

phy_intfc	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

A table that is stored in the address specified by rx_scan_table and as defined in section 6.1.7.3.27.

6.1.7.3.30 Get local POD tracking info (get_loc_pod_trk_info)

The get_loc_pod_trk_info function returns a table of the local tracking information that has accumulated since the last get_loc_pod_trk_info or clr_loc_pod_trk_info function on a specific RF physical interface. The returned table format will be as indicated below.

Syntax:

    * uchar get_loc_pod_trk_info (char-phy_val, uchar * pod_trk_info_tbl)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value: A table that is stored in the address specified by pod_trk_info_tbl.

The scan table is of variable length and indicates all devices found on the channels being scanned.

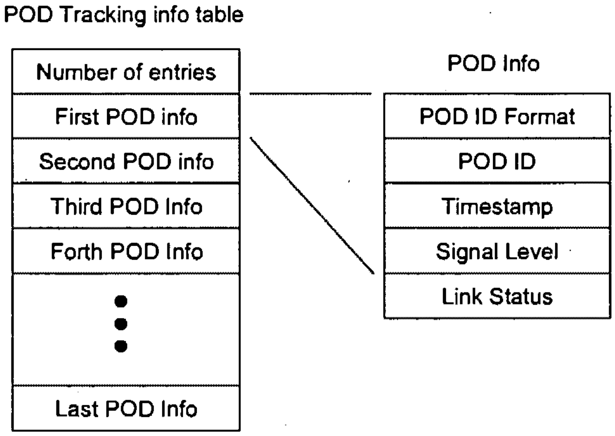


Figure 61: POD Info table diagram

As shown in Figure 61, the pod information table will include a one byte header indicating the number of POD entries attached.. The table will contain an entry for each pod_location_tracking_response returned by a POD.

The pod information block will contain key parameters related to each pod_location_tracking_response received. . Each field in the POD information block is defined below.

6.1.7.3.30.1 POD ID Format (type uchar)

The POD ID Format defines if the POD ID is a short (16 bit) or long (64 bit) POD ID.

dev_id_fmt	Meaning
0	Short Address (16 bits)
1	Long Address (64 bits)

6.1.7.3.30.2 POD ID (type uchar)

The POD ID defines the POD ID detected on this channel for this device. This field will be set to 0 if the device id is unknown. This field will be either a 2 byte field or 8 byte field based on the device ID format field.

dev_id	Meaning
0 to 2^{64}	POD ID value.

6.1.7.3.30.3 Timestamp (type uint)

The timestamp defines when the information was received. The timestamp is the derived from and internal timer that is incremented when the superframe count is incremented. If the superframe count is cleared, the internal timestamp count is also cleared.

timestamp	Meaning
0 to 65535	Internal Timer Count

6.1.7.3.30.4 Signal level (type char)

The signal level defines the current signal level detected while scanning this channel. If a device is found, it reports the signal level during the reception of the devices transmit packet. If no device was found, it defines the average energy detected on the channel.

sig_lvl	Meaning
-127 to +127	Channel signal level.

6.1.7.3.30.5 Link Status (type uchar)

The link status defines the current conditions detected while receiving data from a device. This value determines if errors are found in the receive data. This field will be set to 0 if no device was found.

lnk_stat	Meaning
0	No errors detected

1	Errored packet received.

6.1.7.3.31 Clear local POD tracking info (clr_loc_pod_trk_info)

The clr_loc_pod_trk_info function sends a command to the Proxense chipset to clear the local POD tracking information for a specific RF physical interface.
This function returns a value that indicates pass or fail.

Syntax: uchar clr_loc_pod_trk_info_ (uchar phy_intfc)

phy_intfc	Meaning
0	RF PHY 0
1	RF PHY 1

Returned Value:

Value	Meaning
0	fail
1	pass

6.1.7.3.32 Send POD change channel command (snd_pod_chng_ch_cmd)

The snd_pod_chng_ch_cmd function defines to the Proxense chipset to send a change channel command on a specific RF physical interface to a specific POD.

This function returns a pass or fail indication.

Syntax: char snd_pod_chng_ch_cmd
(uchar phy_val, uint pod_id, uchar ch_num)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID to change channel on

ch_num	Meaning
0 to 15	Destination channel number

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.33 Assign POD temporary short POD ID (asgn_pod_tmp_shrt_id)

The asgn_pod_tmp_shrt_id function defines to the Proxense chipset to send a command on a specific RF physical interface to a specific POD to assign a short ID. The short ID is a 16 bit ID which is used in the place of a long (64 bit) ID.

This function returns a pass or fail indication.

Syntax: char asgn_pod_tmp_shrt_id
(uchar phy_val, uchar[8] pod_long_id, uint pod_shrt_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_long_id	Meaning
0 to 2^{64}	Long POD ID value to change

pod_shrt_id	Meaning
0 to 65535	Assigned temporary POD short ID

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.34 Associate POD (assoc_pod)

The assoc_pod function instructs the Proxense stack and chipset to enter into association mode on a specific RF physical interface to a specific POD. The POD will remain associated until the communications channel is disrupted, or until a disassociate pod function is called.

The associate command causes the POD to go through a full authentication as part of the association process.

This function returns a pass or fail indication.

Syntax: char assoc_pod(uchar phy_val, uint pod_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.35 Disassociate POD (disassoc_pod)

The disassoc_pod function instructs the Proxense stack and chipset to terminate association mode on a specific RF physical interface to a specific POD.

This function returns a pass or fail indication.

Syntax: char assoc_pod(uchar phy_val, uint pod_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.36 Set POD association limits (set_pod_assoc_lmt)

The set_pod_assoc_lmt function instructs the Proxense stack and chipset to communicate on a specific RF physical interface to a specific POD the number of associations a POD may have.

This function returns a pass or fail indication.

Syntax: char set_pod_assoc_lmt(uchar phy_val, uint pod_id, uchar assoc_lmt)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Assoc_lmt	Meaning
0	Illegal value
1 to 15	Number of associations

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.37 Get PODs current number of associations (get_pod_cur_assoc_num)

The get_pod_cur_assoc_num function instructs the Proxense stack and chipset to communicate on a specific RF physical interface to a specific POD and request the current number of associations the POD is involved in.

Syntax: uchar get_pod_cur_assoc_num(uchar phy_val, uint pod_id)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Returned Value:

Assoc_num	Meaning
0 to 15	Number of associations

6.1.7.3.38 Send POD Scan channels command (snd_pod_scn_ch_cmd)

The snd_pod_scn_ch_cmd function instructs the Proxense stack and chipset to communicate on a specific RF physical interface to a specific POD and request the POD to scan a specified number of channels on a specific physical interface for a specified period of time. The POD will scan all channels based on the channel flags return a table indicating the status of each channel.

The response time will be determined based on the number of channels indicated by the channel flags and the number of timeslots to monitor on each channel.

Syntax:

```
        * char_snd_pod_scn_ch_cmd (char phy_intfc, uint pod_id, uint ch_flags,
    uchar num_ts, char * pod_rx_scan_table)
```

phy_intfc	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD·ID

ch_flags	Meaning
Bit 15	Channel 15 Flag
Bit 14	Channel 14 Flag
Bit 13	Channel 13 Flag
Bit 12	Channel 12 Flag
Bit 11	Channel 11 Flag
Bit 10	Channel 10 Flag
Bit 9	Channel 9 Flag
Bit 8	Channel 8 Flag
Bit 7	Channel 7 Flag
Bit 6	Channel 6 Flag
Bit 5	Channel 5 Flag
Bit 4	Channel 4 Flag
Bit 3	Channel 3 Flag
Bit 2	Channel 2 Flag
Bit 1	Channel 1 Flag
Bit 0	Channel 0 Flag

num_ts	Meaning
0 to 255	Defines number of timeslots to scan for on a channel

Returned Value:

A table that is stored in the address specified by pod_rx_scan_table.

The scan table is of variable length and indicates all devices found on the channels being scanned.

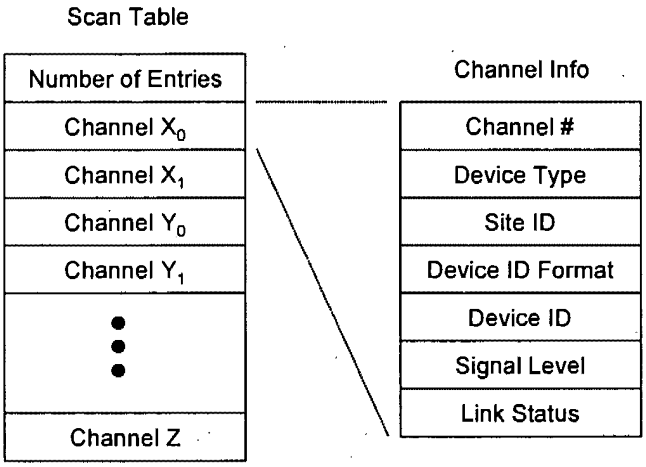


Figure 62: Scan table diagram

As shown in Figure 62, the scan table will include a one byte header indicating the number of channel information blocks attached. Although in most cases a channel will only contain on information block, there are conditions where multiple devices can be operating on a single frequency with cell overlap and the devices cannot see each other, but the device performing the scan can see both. For this reason, the scan table will include all information found on a channel when scanning.

The channel information block will contain key parameters related to what was found on the channel. Each field in the channel information block is defined below.

6.1.7.3.38.1 Channel Number (type char)

The channel number defines the channel that the following information is reported on. The channel number is defined as follows:

chan_num	Meaning
0 to 15	Channel number info is relative to

6.1.7.3.38.2 Device Type (type char)

The device type defines the type of device found on the channel. The following table will define the device types:

dev_typ	Meaning
0	No device found, only energy level is included.
1	Proxense CRDC detected.
2	Proxense RDC detected.
3	Proxense POD detected.
4	Unknown 802.15.4 device detected.
5-255	RFU

6.1.7.3.38.3 Site ID (type uint)

The site id defines the site id detected on this channel for this device. The site id is only valid for CRDC and RDC devices. This field will be set to 0 if the site id is unknown.

site_id	Meaning
0 to 65535	Site ID value.

6.1.7.3.38.4 Device ID Format (type uchar)

The device id format defines the if the device address received is a 16 bit device ID or a 64 bit device ID. This field will default to a short address if the device address is unknown.

dev_id_fmt	Meaning
0	Short Address (16 bits)
1	Long Address (64 bits)

6.1.7.3.38.5 Device ID (type uchar)

The device id defines the device id detected on this channel for this device. This field will be set to 0 if the device id is unknown. This field will be either a 2 byte field or 8 byte field based on the device ID format field.

dev_id	Meaning
0 to 2^{64}	Device ID value.

6.1.7.3.38.6 Signal level (type char)

The signal level defines the current signal level detected while scanning this channel. If a device is found, it reports the signal level during the reception of the devices transmit packet. If no device was found, it defines the average energy detected on the channel.

sig_lvl	Meaning
-127 to +127	Channel signal level.

6.1.7.3.38.7 Link Status (type uchar)

The link status defines the current conditions detected while receiving data from a device. This value determines if errors are found in the receive data. This field will be set to 0 if no device was found.

lnk_stat	Meaning
0	No errors detected
1	Errored packet received.

6.1.7.3.39 Request POD channel assessment (req_pod_ch_asmnt)

The req_pod_ch_asmnt function returns a table of containing an energy assessment of each channel the POD has recently been listening on, on a specific RF physical interface from a specific POD device.

Syntax:

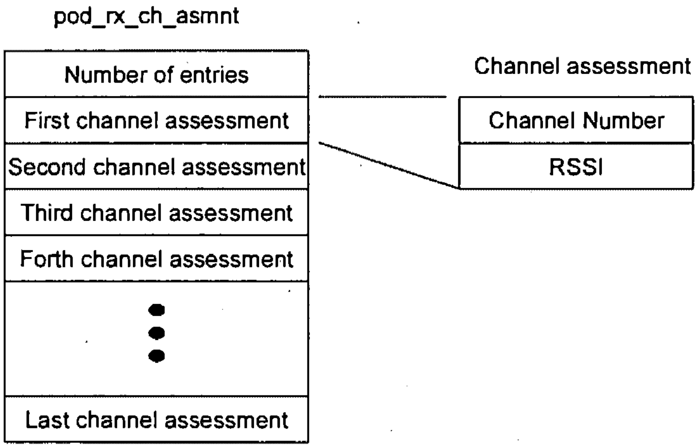
* uchar req_pod_ch_asmnt (char phy_intfc, uint pod_id, char * pod_rx_ch_asmnt)

phy_intfc	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Returned Value:

A table that is stored in the address specified by pod_rx_ch_asmnt.



pod_rx_ch_asmnt

Number of entries
First channel assessment
Second channel assessment
Third channel assessment
Forth channel assessment
•
•
•
Last channel assessment

Channel assessment

Channel Number
RSSI

Figure 63: POD Channel Assessment Table

As shown in Figure 63, the POD channel assessment table will include a one byte header indicating the number of channel assessed. Within each channel assessment entry a channel number and Receiver Signal Strength Indication (RSSI) value will be contained.

6.1.7.3.39.1 Channel Number (type char)

The channel number defines the channel that the following information is reported on. The channel number is defined as follows:

chan_num	Meaning
0 to 15	Channel number info is relative to

6.1.7.3.39.2 RSSI (type char)

The Receiver Signal Strength Indication defines the energy detected on this channel number. The channel number is defined as follows:

rssi_val	Meaning
-127 to +127	Channel RSSI level.

6.1.7.3.40 Request POD connection parameters (req_pod_conn_par)

The req_pod_conn_par function instructs the service layer to send a command on a specific RF physical interface to the POD to get the current connection parameters from the POD. The POD will return a table indicating the current operating status.

Syntax:

*char req_pod_conn_par (uchar phy_val, uint pod_id, char * pod_conn_par)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID

Returned Value:

A table that is stored in the address specified by pod_conn_par.

pod_conn_par
POD RSSI level
POD BER value
POD Power level
POD Mask value
POD Superframe Select
POD Timeslot Select

Figure 64: POD Connection Parameter Table

As shown in Figure 64, the POD connection table contains several specific parameters related to the connection and the current access settings for the POD.

6.1.7.3.40.1 POD RSSI Level

The POD RSSI level is a signed value indicating the received signal strength of the Proxense chipset transmitting this command.

rssi_val	Meaning
-127 to +127	Current receive signal strength indicator

6.1.7.3.40.2 POD BER value

The POD BER value is an unsigned value indicating the average Bit Error Rate measurement of received data since the association was established, or the last request for connection parameters was received.

ber_val	Meaning
0 to 100	An average bit error rate measurement

6.1.7.3.40.3 POD Transmit Power Level

The POD transmit power level value is a signed value indicating the current transmit power setting from the POD.

tx_pwr_val	Meaning
-127 to +127	Current transmit power level

6.1.7.3.40.4 POD mask value

The POD mask value is as specified in section 6.1.7.3.15. The value indicated is the value the POD received during a last beacon reception and is using for superframe and timeslot alignment.

msk_val	Meaning
MSB	Current superframe/timeslot Mask value used (MSB)
LSB	Current superframe/timeslot Mask value used (LSB)

6.1.7.3.40.5 POD superframe select value

The POD superframe select value specifies the superframe number that the POD has computed and will use to communicate on during a location tracking response.

sf_sel	Meaning
MSB	Superframe to communicate on.(MSB)
LSB	Superframe to communicate on.(LSB)

6.1.7.3.40.6 POD timeslot select value

The POD timeslot select value specifies the timeslot number that the POD has computed and will use to communicate on during a location tracking response.

ts_sel	Meaning
0 to sf_len-1	Timeslot to communicate on

6.1.7.3.41 Request Associated POD information (req_assoc_pod_info)

The req_assoc_pod_info function instructs the service layer to send a command on a specific RF physical interface to the POD to get the current pod information. The information includes specific parameters to the pod type and battery indication. The POD will return a table indicating the current pod information.

Syntax: *uchar req_assoc_pod_info (uchar phy_val, uint pod_id, char * pod_info)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID

Returned Value:

A table that is stored in the address specified by pod_info.

pod_info

POD Type
POD Battery Level

Figure 65: POD Information Table

As shown in Figure 65, the POD information table contains the POD type and the battery level.

6.1.7.3.41.1 POD Type

The POD Type indicates the physical POD configuration. The POD may or may not have user interface features..

pod_typ	Meaning
0	No User Interface
1	Single Button interface
2	Biometric interface
3 – 255	RFU

6.1.7.3.41.2 POD Battery Level

The POD battery level indicates a value related to the percentage of charge. The battery level value will range from 0 to 100 indicating the percentage of charge.

pod_bat_lvl	Meaning
0 to 99	Percentage of charge in POD Battery

6.1.7.3.42 Request POD manufacturing parameters (req_pod_mfr_par)

The req_pod_mfr_par function instructs the service layer to send a command on a specific RF physical interface to the POD to get the pods manufacturing parameters. The parameters include manufacturing date and revision information. The POD will return a table indicating this information.

Syntax:

*uchar req_pod_mfr_par (uchar phy_val, uint pod_id, char * pod_mfr_par)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

Pod_id	Meaning
0 to 65535	POD ID

Returned Value:

A table that is stored in the address specified by pod_mfr_par.

pod_mfr_par
Hardware Version
Case Style
Manufacturer ID
Manufacturing date
Software Version

Figure 66: POD Manufacturing Parameters

As shown in Figure 66, the POD information table contains information related to the manufacturing of the POD.

6.1.7.3.42.1 Hardware Version

The hardware version indicates the revision number of the hardware platform and components the POD uses.

hw_ver	Meaning
MSB	Major Revision value
LSB	Minor Revision value

This is a 2 byte field with the high order byte or MSB indicating a major revision number of 0 to 255. The lower order byte indicates a minor revision number. An example of a major and minor revision code would be 1.2 where the major portion is 1 and the minor portion is 2.

6.1.7.3.42.2 Case Style

The case style field indicates the specific style of case used. The case style is an ergonomic definition and does not affect the Proxense protocol implementation.

case_style	Meaning
0	Proxense Style
1-255	RFU

6.1.7.3.42.3 Manufacturer Identification

The manufacturer ID field indicates the assembly house that manufactured the POD.

mfr_id	Meaning
0-255	Manufacturer ID

6.1.7.3.42.4 Manufactured Date

The manufactured date field indicates the date the POD was manufactured. This is a 4 byte field indicating a BCD number for the month, day, and year.

Mfr_date	Meaning
Month	Month(01 to 12)
Day	Day(01 to 31)
Century	Year Centuries (00 to 99)
Years	Years (00 to 99)

An example of a manufacturing code is as follows:

December 30, 2005 = Month = 0x12
Day = 0x30
Century = 0x20
Years = 0x05

6.1.7.3.42.5 Software Version

The software version indicates the revision number of the software the POD uses.

sw_ver	Meaning
MSB	Major Revision value
LSB	Minor Revision value

This is a 2 byte field with the high order byte or MSB indicating a major revision number of 0 to 255. The lower order byte indicates a minor revision number. An example of a major and minor revision code would be 1.2 where the major portion is 1 and the minor portion is 2.

6.1.7.3.43 Send POD Data (snd_pod_data)

The snd_pod_data function instructs the Proxense stack and chipset to transfer data to a specific POD on a specific RF physical interface.

This function returns a pass or fail indication.

Syntax:

```
char snd_pod_data(uchar phy_val, uint pod_id, uchar * tx_pod_data_array)
```

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

tx_pod_data_array	
BC	Byte Count
Data	Contains the number of bytes defined by BC.

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.44 Request POD Data (req_pod_data)

The req_pod_data function instructs the Proxense stack and chipset to send a command to a specific POD on a specific RF physical interface to request data from the POD.

Syntax:

*uchar req_pod_data(uchar phy_val, uint pod_id, uchar * rx_pod_data_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Returned Value:

A table that is stored in the address specified by rx_pod_data_array.

rx_pod_data_array	
BC	Byte Count
Data	Contains the number of bytes defined by BC.

6.1.7.3.45 Send POD Service Provider Data (snd_pod_srvc_prvdr_data)

The snd_pod_srvc_prvdr_data function instructs the Proxense stack and chipset to transfer service provider data to a specific POD on a specific RF physical interface using a temporary session handle.

This function returns a pass or fail indication.

Syntax:

char snd_pod_srvc_prvdr_data(uchar phy_val, uint pod_id, uint session_handle, uchar * tx_pod_srvc_prvdr_data_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

session_handle	Meaning
0 to 65535	A temporary handle assigned for service provider transactions

tx_pod_srvc_prvdr_data_array	
BC	Byte Count
Data	Contains the number of bytes defined by BC.

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.46 Request POD Service Provider Data (req_pod_srvc_prvdr_data)

The req_pod_srvc_prvdr_data function instructs the Proxense stack and chipset to send a command to a specific POD on a specific RF physical interface with a specific session handle to request service provider data from the POD.

Syntax:

*uchar req_pod_srvc_prvdr_data(uchar phy_val, uint pod_id, uint session_handle, uchar * rx_pod_srvc_prvdr_data_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

session_handle	Meaning
0 to 65535	A temporary handle assigned for service provider transactions

Returned Value:

A table that is stored in the address specified by rx_podsrvc_prvdr_data_array.

rx_pod_srvc_prvdr_data_array	
BC	Byte Count
Data	Contains the number of bytes defined by BC.

6.1.7.3.47 Unlock POD Service Provider Area (unlock_pod_svc_prvdr_area)

The unlock_pod_srvc_prvdr_area function instructs the Proxense stack and chipset to send a command to the POD to unlock a specific Service Provider Data Area.

This function returns a session handle which is required to write or read data to/from the specific service provider area in the POD.

Syntax:

uint unlock_pod_srvc_prvdr_area(uchar phy_val, uint pod_id, uchar * srvc_prvdr_unlock_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

srvc_prvdr_unlock_array		
Name	Bits	Description
SPSCID	63:0	Service Provider Secret ID (8 bytes)
SPSCK	63:0	Service Provider Secret Key (8 Bytes)

Returned Value:

Value	Meaning
0	Fail
1 to 65535	session_handle

6.1.7.3.48 Lock POD Service Provider Area (lock_pod_svc_prvdr_area)

The lock_pod_srvc_prvdr_area function instructs the Proxense stack and chipset to send a command to the POD to lock a specific Service Provider Data Area.

This function returns a response indicating the lock was successful.

Syntax:

uint lock_pod_srvc_prvdr_area(uchar phy_val, uint pod_id, uint session_handle)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

Value	Meaning
0	Reserved
1 to 65535	session_handle

Returned Value:

Value	Meaning
0	Fail
1 to 65535	session_handle

6.1.7.3.49 Initialize POD Service Provider Area (init_pod_svc_prvdr_area)

The init_pod_srvc_prvdr_area function instructs the Proxense stack and chipset to send a command to the POD to initialize a new Service Provider Data Area. In order for the this function to complete, the service provider must issue a unlock_pod_srvc_prvdr_area to guarantee the area is accessible.

This function returns a value indicating pass or fail.

Syntax:

char init_pod_srvc_prvdr_area(uchar phy_val, uint pod_id, uchar mem_typ, uchar * srvc_prvdr_init_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

mem_typ	Meaning
0	temporary
1	permanent

srvc_prvdr_init_array		
Name	Bits	Description
SPPID	15:0	Service Provider POD ID
SPSTID	15:0	Service Provider Site ID
SPSCID	63:0	Service Provider Secret ID (8 bytes)
SPSK	63:0	Service Provider Secret Key (8 Bytes)
SPDBC	7:0	Service Provider Data Byte Count
SPDATA	(SPDBC*8)-1:0	Service Provider Data Contents

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.50 Release POD Service Provider Area (rel_pod_svc_prvdr_area)

The rel_pod_srvc_prvdr_area function instructs the Proxense stack and chipset to send a command to the POD to release a specific Service Provider Area from the PODs internal memory. Once the service provider area is released, the area becomes available for another service provider.

Note: It is not required to release information stored in temporary memory, but may be released at the service provider's discretion.

This function returns a value indicating pass or fail.

Syntax:

char rel_pod_srvc_prvdr_area(uchar phy_val, uint pod_id, uchar * srvc_prvdr_unlock_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

srvc_prvdr_unlock_array		
Name	Bits	Description
SPSCID	63:0	Service Provider Secret ID (8 bytes)
SPSCK	63:0	Service Provider Secret Key (8 Bytes)

Returned Value:

Value	Meaning
-1	Fail
0	Pass

6.1.7.3.51 Initialize POD Primary Service Provider Area (init_pod_prim_svc_prvdr_area)

The init_pod_prim_srvc_prvdr_area function instructs the Proxense stack and chipset to send a command to the POD to initialize the primary Service Provider Data Area. The issuer of the POD may perform this function and will allow additional settings in the POD as to whether other service providers may store temporary or permanent information in the PODs service provider areas. In order for this function to complete, the service provider must issue a unlock_pod_srvc_prvdr_area to guarantee the area is accessible.

This function returns a value indicating pass or fail.

Syntax:

char init_pod_prim_srvc_prvdr_area(uchar phy_val, uint pod_id, uchar mem_typ, uchar temp_en, uchar perm_num, uchar perm1_attr, uchar perm2_attr, uchar perm3_attr, uchar perm4_attr, uchar * srvc_prvdr_init_array)

phy_val	Meaning
0	RF PHY 0
1	RF PHY 1

pod_id	Meaning
0 to 65535	POD ID

6.1.7.3.51.1 Temporary Memory Enable

The temporary memory enable field defines if the temporary storage area may be used by any service provider wanting to store memory in temporary memory.

temp_en	Meaning
0	Disallow temporary service provider storage
1	Allow temporary service provider storage

6.1.7.3.51.2 Permanent Memory Number

The permanent memory number field defines if the number of permanent memory storage areas available for service providers to store their service provider profile in.

perm_num	Meaning
0	Disallow permanent service provider storage (except this provider)
1-255	Allow permanent storage for this number of service providers

6.1.7.3.51.3 Permanent X attribute

The permanent X attribute contains 4 fields that define how specific serive providers can store their profile in the permanent memory. Each of the 4 fields can be used to govern how up to 4 service providers can store their data. The lower 7 bits of the field define the upper 7 bits of a service provider ID. The most significant bit defines whether the service provider specified by the lower 7 bits can write their profile one time, or not at all. A value of 0x00 in any field implies that no attribute is specified. A value of 0xFF implies that each service provider may only write their data into permanent memory one time.

permx_attr	Meaning
0x00	No attribute specified
0x01-0x7E	Disallow service provider storage for specified value.
0x7F	Disallow all service providers from storing.
0x80	Reserved
0x81 – 0xFE	Allow a one time service provider storage defined by 7 least significant bits.
0xFF	Allow all service providers to store one time.

6.1.7.3.51.4 Memory Type

The memory type field indicates where the primary service provider will write this profile.

mem_typ	Meaning
0	temporary
1	permanent

srvc_prvdr_init_array		
Name	Bits	Description
SPPID	15:0	Service Provider POD ID
SPSTID	15:0	Service Provider Site ID
SPSCID	63:0	Service Provider Secret ID (8 bytes)
SPSK	63:0	Service Provider Secret Key (8 Bytes)
SPDBC	7:0	Service Provider Data Byte Count
SPDATA	(SPDBC*8)-1:0	Service Provider Data Contents

Returned Value:

Value	Meaning
-1	Fail
0	Pass

Annex A. Service Provider Area Access

A Service Provider by the permission of Proxense, will have the ability to store secure information into the POD in a Service Provider memory area. The data is stored as binary data and the Proxense POD does not perform any manipulation on the data. The Service Provider may perform any security measures they desire to the data prior to sending it to the secure area.

The Service Provider area is protected for each service provider by a secret service provider ID and a secret Key. The service provider must be capable of passing this information to the POD during an unlock POD service provider area to gain access to their information.

If an unauthorized person attempts to unlock the service provider area and does not have the secret ID or Key, they will get 3 attempts before the POD will disallow any further interaction to the service provider area until a different site id is found.

Annex B. CRDC Synchronization

It is strongly advised that CRDCs are synchronized to each other to preserve the POD battery life, but it is not a specific requirement.

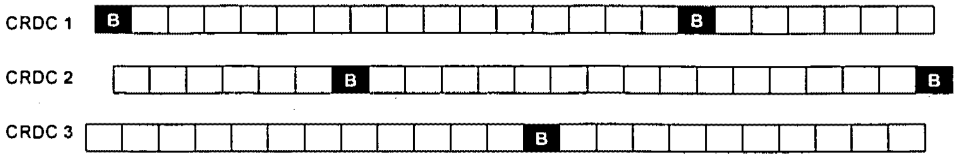


Figure 67: Unsynchronized CRDC Timing

Figure 67 illustrates an arbitrary alignment of the Superframe and Beacon a specific POD will listen for a CRDC.

Five options of synchronizing CRDC cells are possible.

1. Wireline synchronization between CRDCs.
2. Wireline synchronization through connections located at the Central Server.
3. Synchronization via the wireline communications channel implemented at the Central Server.
4. A master CRDC that has ubiquitous coverage over all CRDCs in the casino.
5. CRDC to CRDC cell overlap where each CRDC synchronizes to the next CRDC.

B.1 Option 1 – wireline synchronization between CRDCs

Based on option 1, each CRDC can be daisy chained to the next CRDC, where the first CRDC is the master and provides the master timing reference to the next CRDC where that CRDC has a wireline repeater for the next CRDC in series as well as acts as a slave and recovers the master timing reference to adjust its over-the-air timing.

Option 1 may also be configured into a star network provided an addition driver distribution circuit is provided to allow other CRDCs to be attached in a star configuration and/or a daisy chain.

The implication of this approach is the fact that additional wiring must be run between CRDCs and most likely that wiring will be difficult and timely to install.

B.2 Option 2 – Wireline synchronization through connections located at the Central Server

Based on option 2, where the units are synchronized through connection located at the Central Server. This may be a viable approach given at least one additional wire pair is available in the communications cable connecting the Central Server to the CRDC. By locating a clock source near the Central Server (but not implemented in the Central Server), and a driver distribution circuit, all CRDCs can be connected through the driver distribution circuit in a star configuration.

The implications of this approach are an additional timing device will need to be developed and one additional wire pair must be available. This approach should be fairly easy to install since the connections are centrally located.

B.3 Option 3 - Synchronization via the wireline communications channel implemented at the Central Server

Option 3 implies that the Central Server will be capable of communicating with the CRDCs on a precise interval. This will place a timing constraint on the Central Processor and may not be viable for the casino operator.

B.4 Option 4 - A master CRDC that has ubiquitous coverage over all CRDCs in the casino

Option 4 implies that if a master CRDC has ubiquitous coverage in the casino, then only the one CRDC is required.

B.5 Option 5 - CRDC to CRDC cell overlap where each CRDC synchronizes to the next CRDC

Option 5 allows each CRDC cell to be large enough to be detected by adjacent CRDCs. If a CRDC has a dual phy, each CRDC could lock to the next CRDC.
The disadvantage of this scheme, is once the cell sizes grow large enough to overlap the adjacent CRDC, channel reutilization becomes more limited. To explain further, Each CRDC can output a CRDC beacon of which the next CRDC can receive. This sound efficient since every CRDC transmits on a separate channel anyway. But you must take into account that the RDCs within a CRDC cell also have fewer channels to communicate to the POD on, since more of the channels are now occupied with adjacent CRDC beacon transmission.

Synchronization conclusion

Annex C. POD timeslot and superframe assignment and distribution

A basic understanding of timeslot and superframe assignment is given in section 5.2.1.2.3. The assignment may be broken down further into preferred players and generic players. The casino operator may choose to allow a prefer player to access the system at a higher repetition rate giving the player a guaranteed timeslot without interference by using a low capacity assigned superframe and time slot.

In order for the casino operator to do this, the Service Providers POD ID becomes an essential part of the configuration.

There are several components required to determine when POD access can occur in the Beacon framing structure. As previously mentioned, the Beacon is always in timeslot 0, so timeslot 0 is always prohibited.

The main components that determine where in the Beacon frame a POD can access the system are:

- Network Format / Timeslot Select – determines odd/even slot assignment
- Superframe Count - defines current superframe count
- Timeslot Count – defines current timeslot count
- POD_SF_TS_Msk – defines which superframe and timeslot bits to consider for access.
- Service Provider POD ID – used as part of the computation for this POD

Although this has already been explained on a basic level, it has not been explained from a distribution and specific assignment standpoint.

C.1 Timeslot assignment

First the timeslot assignment will be explained. The specific fields associated with timeslot assignment are shown in Table 12.

Proxense Network Format		
Network Type	2	Multi-cell CRDC coordinated
Beacon Source	0	CRDC transmits beacon
Broadcast Flag	1	Broadcast
Timeslot Select	3	POD uses odd timeslots for access
POD_SF_TS Msk	0x0007	Superframe masked and all timeslot bits enabled

Table 12: Example timeslot assignment

As shown, Timeslot Select is set to allow the POD to only respond in odd timeslots on the beacon channel. Also shown is all superframe bits are masked and only the timeslot bits are enabled for comparison against the Service Provider POD ID.

By defining the Service Provider POD ID value, the service provider can define where the POD will be located within the timeslots.

If 2 PODs were added to the system and one POD had a Service Provider POD ID of 0x0012 and the second POD had a Service Provider POD ID or 0x007E the following timeslots would be occupied.

Timeslot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
POD 1 (0x0102)	B					T	R									
POD 2 (0x0206)	B													T	R	

Table 13: Timeslot allocation for 2 PODs

As shown in Table 13, Timeslot 0 indicates a Beacon timeslot. Timeslot 5 is the transmit timeslot for POD1 and timeslot 6 is the response from an RDC if the RDC wants to communicate back to POD1. Timeslot 13 is the transmit timeslot for POD2 and timeslot 14 is the response from an RDC if the RDC wants to communicate back to POD2.

Remember that the timeslot selected for each POD is based on shifting the 2 lsbs of the timeslot count by one and adding a 1 or 0 to the lsb for determining odd or even timeslots.

Although it is not likely a single superframe will be used in the system, the example illustrates how the Service Provider POD ID defines where the POD may communicate to an RDC on the beacon channel.

Note: A POD cannot have it's least significant values set to 7, since this would cause the POD to respond in timeslot 15 and a RDC would not be able to respond in timeslot 0 since it is designated as the Beacon timeslot.

C.2 Superframe assignment

Superframe assignment follows similar principles as timeslot assignment but at the superframe level utilizing the superframe count, mask, and Service Provider POD ID bits associated with the superframe.

There is no superframe format specifier to determine if odd or even superframes are in use, but only the superframe count.

Proxense Network Format		
Network Type	2	Multi-cell CRDC coordinated
Beacon Source	0	CRDC transmits beacon
Broadcast Flag	1	Broadcast
Timeslot Select	3	POD uses odd timeslots for access
POD_SF_TS Msk	0x007F	Superframe 4 lsbs and all timeslot bits enabled

Table 14: Example timeslot assignment

As shown, the superframe bits enable only the 4 lsbs of the superframe count and all of the timeslot bits are enabled for comparison against the Service Provider POD ID.

Again, by defining the Service Provider POD ID value, the service provider can define where the POD will be located within superframes and timeslots.

Using the same Service Provider POD IDs as in the timeslot example, the same timeslot for each POD would be occupied for each of the following superframes.

Superframe	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
POD 1 (0x0102)			X													
POD 2 (0x0206)																X

Table 15: Superframe allocation for 2 PODs

As shown in Table 15, superframe 2 is the transmit superframe for POD1 and superframe 15 is the transmit superframe for POD2. Again, the RDC will respond in these superframes on the next even timeslot following the POD transmission, if the RDC wants to communicate with that POD.

Remember that the superframe selected for each POD is based on the lsb being located in bit position 3 of the Service Provider POD ID.

This example illustrated how a POD can be positioned anywhere within the superframe structure.

C.3 Service Provider POD Mask

In addition to a site mask that is sent by the C-Beacon, it is possible for an additional mask to be configured within each POD. This is a Service Provider POD mask that is exclusive-ored with the POD_SF_TS_Msk value received by the beacon.

By allowing the exclusive-or of the POD_SF_TS_Msk, a specific POD can be configured to have access on other timeslots that a normal POD wouldn't have access to at the same frequency.

Referring to the allocation in Table 15, if the POD 1 with the ID 0x0012 contained a Service Provider POD Mask of 0x0060, then this value would be exclusive-ored with the POD_SF_TS_Msk of 0x007F resulting in a mask value of 0x001F. This in turn would be used for the mask of the Service Provider POD ID and the Superframe and timeslot counts.

Superframe	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
POD 1 (0x0102)			X		X		X		X		X		X		X	
POD 2 (0x0206)																X

Table 16: Superframe allocation using internal POD mask for POD1

As shown in Table 16, POD 1 would then awake on every other superframe and provide a response where POD 2 would only respond on Frame 15. This would give the player with POD 1 a faster system response time and the Central Server a higher tracking rate, while also decreasing battery efficiency. The player carrying POD 2 would also get a response, but at a slower rate.

Although the operator would in most cases not set the system up with such a high response rate, this example shows how the internal POD mask could operate.

C.4 Conclusion

As described in the above sections it is possible to control a PODs response to specific timeslots and superframe, or multiple timeslots and superframes. The system can control the rate by adjusting the C-superframe and Superframe lengths and by the assigned Service Provider POD ID. The operator can also increase a users access and response to the system by providing an internal POD mask field.

Besides being able to control where each POD is located throughout the superframe structure, the operator has the option of assigning the PODs in groups if desired.

PATENT APPLICATION SERIAL NO. ____________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

```
01/23/2006 FMETEKI1 00000091 60760362

01 FC:2005                        100.00 OP
02 FC:2085                        250.00 OP
```

PTO-1556
(5/87)

*U.S. Government Printing Office: 2002 — 489-267/69033